arXiv:1112.6384v1 [cs.CL] 29 Dec 2011

Proof nets for the Lambek-Grishin calculus

Michael Moortgat[1] and Richard Moot[2] ⋆

[1] Utrecht Institute of Linguistics OTS
Trans 10
3512 JK Utrecht, Netherlands
M.J.Moortgat@uu.nl,
[2] CNRS, Université de Bordeaux
LaBRI, 351 cours de la Libération
33400 Talence, France
moot@labri.fr

Abstract. Grishin's generalization of Lambek's Syntactic Calculus combines a non-commutative multiplicative conjunction and its residuals (product, left and right division) with a dual family: multiplicative disjunction, right and left difference. Interaction between these two families takes the form of linear distributivity principles. We study proof nets for **LG** and the correspondence between these nets and unfocused and focused versions of its sequent calculus.

1 Background, motivation

In his two seminal papers (Lambek 1958; Lambek 1961), Jim Lambek introduced the 'parsing as deduction' method in linguistics: the traditional parts of speech (noun, verb, adverb, determiner, etc) are replaced by logical formulas — types if one takes the computational view; the judgement whether an expression is well-formed is the outcome of a process of logical deduction, or, reading formulas as types, a computation in the type calculus.

$$\underset{\text{time}}{np} \otimes \underset{\text{flies}}{(np\backslash s)} \otimes \underset{\text{like}}{(((np\backslash s)\backslash(np\backslash s))/np)} \otimes \underset{\text{an}}{(np/n)} \otimes \underset{\text{arrow}}{n} \rightarrow s \tag{1}$$

What is the precise nature of grammatical composition, the $\otimes$ operation in the example above? The '58 and '61 papers present two views on this: in the '58 paper, types are assigned to *strings* of words, in the '61 paper, they are assigned to *phrases*, bracketed strings, with a grouping into constituents. The Syntactic Calculus, under the latter view, is extremely simple. The derivability relation between types is given by the preorder laws (2) and the residuation principles of (3).

$$A \rightarrow A \quad ; \quad \text{from } A \rightarrow B \text{ and } B \rightarrow C \text{ infer } A \rightarrow C \tag{2}$$

⋆ Draft of a chapter in E. Grefenstette, C. Heunen, and M. Sadrzadeh (eds.) 'Compositional methods in Physics and Linguistics', OUP, to appear. We thank Arno Bastenhof for helpful comments on an earlier version.

$$A \to C/B \quad \text{iff} \quad A \otimes B \to C \quad \text{iff} \quad B \to A\backslash C \tag{3}$$

To obtain the '58 view, one adds the *non-logical axioms* of (4), attributing associativity properties to the $\otimes$ operation.

$$(A \otimes B) \otimes C \to A \otimes (B \otimes C) \qquad ; \qquad A \otimes (B \otimes C) \to (A \otimes B) \otimes C \tag{4}$$

The Syntactic Calculus in its two incarnations — the basic system **NL** given by (2) and (3) and the associative variant **L** which adds the postulates of (4) — recognizes only context-free languages. It is well known that to capture the dependencies that occur in natural languages, one needs expressivity beyond context-free. Here are some characteristic patterns from formal language theory that can be seen as suitable idealizations of phenomena that occur in the wild.

$$\begin{aligned} \text{copying: } & \{w^2 \mid w \in \{a,b\}^+\} \\ \text{counting dependencies: } & \{a^n b^n c^n \mid n > 0\} \\ \text{crossed dependencies: } & \{a^n b^m c^n d^m \mid n, m > 0\} \end{aligned} \tag{5}$$

In the tradition of extended rewriting systems, there is a large group of grammar formalisms that handle these and related patterns gracefully: Tree Adjoining Grammars, Linear Indexed Grammars, Combinatory Categorial Grammars, Minimalist Grammars, Multiple Context Free Grammars, ... (Kallmeyer 2010). Also in the Lambek tradition, extended type-logical systems have been proposed with expressive power beyond context-free: multimodal grammars (Morrill 1994; Moortgat 1996), discontinuous calculi (Morrill, Fadda, and Valentin 2007), etc. These extensions, as well as the original Lambek systems, respect an "intuitionistic" restriction: in a sequent presentation, derivability is seen as a relation between (a structured configuration of) hypotheses $A_1, \ldots, A_n$ and a *single* conclusion B. In a paper antedating Linear Logic by a couple of years, Grishin (1983) proposes a generalization of the Lambek calculus which removes this intuitionistic restriction. Linguistic application of Grishin's ideas is fairly recent. In the present paper, we study the system presented in (Moortgat 2009), which we'll refer to as **LG**.

1.1 Dual residuation principles, linear distributivities

In **LG** the inventory of type-forming operations is doubled: in addition to the familiar operators $\otimes, \backslash, /$ (product, left and right division), we find a dual family $\oplus, \oslash, \otimes$: coproduct, right and left difference.

$$\begin{array}{lll} A, B ::= p \mid & & \text{atoms: } s,\ np,\ \ldots \\ \quad A \otimes B \mid B\backslash A \mid A/B \mid & & \text{product, left vs right division} \\ \quad A \oplus B \mid A \oslash B \mid B \otimes A & & \text{coproduct, right vs left difference} \end{array} \tag{6}$$

Some clarification about the notation: we follow (Lambek 1993) in writing $\oplus$ for the coproduct, which is a multiplicative operation, like $\otimes$. We read $B\backslash A$ as 'B under A', A/B as 'A over B', $B \otimes A$ as 'B from A' and $A \oslash B$ as 'A less B'.

For the difference operations, then, the quantity that is subtracted is under the circled (back)slash, just as we have the denominator under the (back)slash in the case of left and right division types. In a formulas-as-types spirit, we will feel free to refer to the division operations as implications, and to the difference operations as co-implications.

Dual residuation principles The most basic version of **LG** is the symmetric generalization of **NL**, which means that to (2) and (3) we add the dual residuation principles of (7).

$$B \oslash C \to A \quad \text{iff} \quad C \to B \oplus A \quad \text{iff} \quad C \oslash A \to B \tag{7}$$

To get a feeling for the consequences of the preorder laws (2) and the (dual) residuation principles (3) and (7), here are some characteristic theorems and derived rules of inference. First, the *compositions* of the product and division operations, and of the co-product and difference operation give rise to the expanding and contracting patterns of (8). The rows here are related by a left-right symmetry; the columns by arrow reversal.

$$\begin{array}{cc} A \otimes (A\backslash B) \to B \to A\backslash(A \otimes B) & (B/A) \otimes A \to B \to (B \otimes A)/A \\ (B \oplus A) \oslash A \to B \to (B \oslash A) \oplus A & A \oslash (A \oplus B) \to B \to A \oplus (A \oslash B) \end{array} \tag{8}$$

Secondly, one can show that the type-forming operations have the monotonicity properties summarized in the following schema, where $\uparrow$ ($\downarrow$) is an isotone (antitone) position:

$$(\uparrow \otimes \uparrow), (\uparrow / \downarrow), (\downarrow \backslash \uparrow), (\uparrow \oplus \uparrow), (\uparrow \oslash \downarrow), (\downarrow \oslash \uparrow)$$

In other words, the following inference rules are valid.

$$\frac{A' \to A \quad B' \to B}{A' \otimes B' \to A \otimes B} \qquad \frac{A \to A' \quad B \to B'}{A \oplus B \to A' \oplus B'} \tag{9}$$

$$\frac{A' \to A \quad B \to B'}{A\backslash B \to A'\backslash B'} \quad \frac{A' \to A \quad B \to B'}{A' \oslash B' \to A \oslash B} \quad \frac{A' \to A \quad B \to B'}{B/A \to B'/A'} \quad \frac{A' \to A \quad B \to B'}{B' \oslash A' \to B \oslash A} \tag{10}$$

Interaction: distributivity principles As we saw above, one could extend the inferential capabilities of this minimal system by adding postulates of associativity and/or commutativity for $\otimes$ and $\oplus$. From a substructural perspective, each of these options destroys structure-sensitivity for a particular dimension of grammatical organization: word order in the case of commutativity, constituent structure in the case of associativity. In **LG** there is an alternative which leaves the sensitivity for linear order and phrasal structure intact: instead of considering structural options for the individual $\otimes$ and $\oplus$ families, one can consider

interaction principles for the communication between them. We will consider the following group.

$$\begin{array}{ll} (A \oslash B) \otimes C \to A \oslash (B \otimes C) & C \otimes (B \oslash A) \to (C \otimes B) \oslash A \\ C \otimes (A \oslash B) \to A \oslash (C \otimes B) & (B \oslash A) \otimes C \to (B \otimes C) \oslash A \end{array} \tag{11}$$

These postulates have come to be called *linear distributivity principles* (e.g. (Cockett and Seely 1996)): linear, because they respect resources (no material gets copied). Moot (2007) models the adjunction operation of Tree Adjoining Grammars using the interaction principles of (11) and shows how through this modeling the mildly context-sensitive patterns of (5) can be obtained within **LG**.

1.2 Arrows: LG as a deductive system

In his (Lambek 1988), Lambek studies the Syntactic Calculus from a categorical perspective. Types are seen as the objects of a category and one studies morphisms between these objects, arrows $f : A \longrightarrow B$. For each A, there is an identity arrow 1_A. Then there are inference rules to produce new arrows from arrows already obtained. Among these is the composition $g \circ f$, defined when $dom(g) = cod(f)$. Composition is associative, i.e. one has the equation $f \circ (g \circ h) = (f \circ g) \circ h$. Also, $f \circ 1_A = f = 1_B \circ f$, where $f : A \longrightarrow B$.

$$1_A : A \longrightarrow A \qquad \frac{f : A \longrightarrow B \quad g : B \longrightarrow C}{g \circ f : A \longrightarrow C} \tag{12}$$

In this paper, we will not pursue the categorical interpretation of **LG**: our emphasis in the following sections is on the *sequent* calculus for this logic, the term language coding sequent proofs, and the correspondence between these proofs and proof nets. Our aim in this section is simply to have a handy language for naming proofs in the deductive presentation, and to use this in §2.1 to establish the equivalence between the deductive and the sequent presentations.

To obtain **aLG**, one adds to (12) further rules of inference for the residuation principles and their duals. (Omitting type subscripts $\triangleright_{A,B,C} f$ for legibility...)

$$\frac{f : A \otimes B \longrightarrow C}{\triangleright f : A \longrightarrow C/B} \qquad \frac{f : A \otimes B \longrightarrow C}{\triangleleft f : B \longrightarrow A\backslash C} \tag{13}$$

$$\frac{g : A \longrightarrow C/B}{\triangleright^{-1} g : A \otimes B \longrightarrow C} \qquad \frac{g : B \longrightarrow A\backslash C}{\triangleleft^{-1} g : A \otimes B \longrightarrow C} \tag{14}$$

$$\frac{f : C \longrightarrow B \oplus A}{\blacktriangleleft f : B \oslash C \longrightarrow A} \qquad \frac{f : C \longrightarrow B \oplus A}{\blacktriangleright f : C \oslash A \longrightarrow B} \tag{15}$$

$$\frac{g : B \oslash C \longrightarrow A}{\blacktriangleleft^{-1} g : C \longrightarrow B \oplus A} \qquad \frac{g : C \oslash A \longrightarrow B}{\blacktriangleright^{-1} g : C \longrightarrow B \oplus A} \tag{16}$$

As remarked above, the Lambek-Grishin calculus exhibits two involutive symmetries, at the level of types and proofs: a left-right symmetry $\cdot^\natural$ and an arrow reversing symmetry $\cdot^\dagger$ such that

$$f^\natural : A^\natural \longrightarrow B^\natural \quad \text{iff} \quad f : A \longrightarrow B \quad \text{iff} \quad f^\dagger : B^\dagger \longrightarrow A^\dagger \tag{17}$$

with, on the type level, the translation tables below (abbreviating a long list of defining equations $(A \otimes B)^\natural \doteq B^\natural \otimes A^\natural, (B \otimes A)^\natural \doteq A^\natural \otimes B^\natural, \ldots$)

$$\begin{array}{|cccc|}\hline A \otimes B & A/B & A \oplus B & A \oslash B \\ \hline\hline B \otimes A & B\backslash A & B \oplus A & B \oslash A \\ \hline\end{array}\ {}^\natural \qquad \begin{array}{|ccc|}\hline A/B & A \otimes B & B\backslash A \\ \hline\hline B \oslash A & B \oplus A & A \oslash B \\ \hline\end{array}\ {}^\dagger$$

and on the level of proofs $(1_A)^\natural = 1_{A^\natural}$, $(g \circ f)^\natural = g^\natural \circ f^\natural$, $(1_A)^\dagger = 1_{A^\dagger}$, $(g \circ f)^\dagger = f^\dagger \circ g^\dagger$, and the list of defining equations $(\triangleleft f)^\natural \doteq \triangleright f^\natural$, $(\triangleleft f)^\dagger \doteq\blacktriangleright f^\dagger$, ... corresponding to the translation tables above.

The distributivity principles, in **aLG**, take the form of extra axioms (primitive arrows). Below arrows $\mathbf{d}$, $\mathbf{b}$ for the interaction between $\oslash$ and $\otimes$. For the left-right symmetric pair $\mathbf{d}^\natural$, $\mathbf{q}^\natural$ we write $\mathbf{b}$, $\mathbf{p}$

$$\begin{array}{l}\mathbf{d}_{A,B,C} : (A \oslash B) \otimes C \longrightarrow A \oslash (B \otimes C) \\ \mathbf{q}_{A,B,C} : C \otimes (A \oslash B) \longrightarrow A \oslash (C \otimes B)\end{array} \tag{18}$$

To establish the equivalence between **aLG** and the sequent calculus **sLG**, to be discussed in the next section, we will use the fact that the monotonicity rules are derived rules of inference of **aLG**. For example, f/g can be defined as in (19) below.

$$\begin{array}{c}\dfrac{f : A \longrightarrow A' \quad g : B \longrightarrow B'}{f/g : A/B' \longrightarrow A'/B} \\ f/g \doteq (\triangleright(f \circ (\triangleright^{-1} 1_{A/B})))) \circ (\triangleright \triangleleft^{-1} ((\triangleleft \triangleright^{-1} 1_{A/B'}) \circ g))\end{array} \tag{19}$$

Similarly, for the distributivity postulates, we will rely on a rule form, which for $\mathbf{d}$ would be

$$\frac{B \otimes C \to A \oplus D}{A \oslash B \to D/C} \tag{20}$$

The inference rule (20) is derived as shown in (21).

$$\dfrac{\mathbf{d}_{A,B,C} : (A \oslash B) \otimes C \longrightarrow A \oslash (B \otimes C) \quad \dfrac{f : B \otimes C \longrightarrow A \oplus D}{\blacktriangleleft f : A \oslash (B \otimes C) \longrightarrow D}}{\dfrac{(\blacktriangleleft f) \circ \mathbf{d}_{A,B,C} : (A \oslash B) \otimes C \longrightarrow D}{\triangleright((\blacktriangleleft f) \circ \mathbf{d}_{A,B,C}) : A \oslash B \longrightarrow D/C}} \tag{21}$$

2 Display sequent calculus and proof nets

Is there a decision procedure to determine whether $A \to B$ holds? In the presence of *expanding* patterns as we saw them in (8), this is not immediately clear. For

the language with $/, \otimes, \backslash$, the key result of Lambek's original papers was to establish decidability by applying Gentzen's method: the Syntactic Calculus is recast as a sequent calculus; for the sequent presentation one then shows that the Cut rule (the sequent form of transitivity) is admissible; backward-chaining, cut-free proof search then yields the desired decision procedure.

In §2.1 below, we work through a similar agenda for **LG**. We introduce **sLG**, a sequent system for the Lambek-Grishin calculus in the style of Display Logic (Goré 1997), and show that it is equivalent to **aLG**. The sequent presentation enjoys Cut Elimination; decidability follows. Sequent proof search, though decidable, remains suboptimal in that it allows a great many derivations for what in effect one would like to consider as 'the same' proof. In §2.2, we introduce proof nets for **LG**, and show how these nets remove the spurious forms of nondeterminism of sequent proof search.

2.1 sLG: display sequent calculus

The arrows of **aLG** are morphisms between *types*. In the sequent calculus, derivability is a relation between *structures* built from types. We will present the sequent calculus for **LG** in the format of a Display Logic (see (Goré 1997) for a comprehensive display logical view on the substructural landscape). The characteristic feature of Display Logic is that for every logical connective, there is a corresponding structural connective. We use the same symbols for the logical operations and their structural counterparts; structural operations are marked off by centerdots. Below the grammar for input (sequent left hand side), and output structures (sequent rhs).

$$\begin{array}{lcl} \mathcal{I} & ::= & \mathcal{F} \mid \mathcal{I} \cdot \otimes \cdot \mathcal{I} \mid \mathcal{I} \cdot \oslash \cdot \mathcal{O} \mid \mathcal{O} \cdot \oslash \cdot \mathcal{I} \\ \mathcal{O} & ::= & \mathcal{F} \mid \mathcal{O} \cdot \oplus \cdot \mathcal{O} \mid \mathcal{I} \cdot \backslash \cdot \mathcal{O} \mid \mathcal{O} \cdot / \cdot \mathcal{I} \end{array}$$

The rules of **sLG** come in three groups: the identity group (Axiom, Cut), the structural group (Display Postulates, Distributivity Postulates), and the logical group (left and right introduction rules for the logical connectives). Variables X, Y, Z in these rules range over structures, input or output, depending on whether they appear left or right of the sequent arrow.

Axiom, Cut

$$\frac{}{A \Rightarrow A}\ \text{Ax} \qquad \frac{X \Rightarrow A \quad A \Rightarrow Y}{X \Rightarrow Y}\ \text{Cut} \tag{22}$$

Display postulates The (dual) residuation principles are formulated at the structural level. These rules ensure that any formula constituent of a sequent can be displayed as the single occupant of the sequent lhs or rhs—hence the name.

$$\frac{\dfrac{X \Rightarrow Z \cdot / \cdot Y}{X \cdot \otimes \cdot Y \Rightarrow Z}\, rp}{Y \Rightarrow X \cdot \setminus \cdot Z}\, rp \qquad \frac{\dfrac{Y \cdot \oslash \cdot Z \Rightarrow X}{Z \Rightarrow Y \cdot \oplus \cdot X}\, drp}{Z \cdot \oslash \cdot X \Rightarrow Y}\, drp \tag{23}$$

Distributivity postulates The linear distributivities motivate the choice for a *display* sequent calculus. The distributivity postulates, in their rule form of (20), in the sequent format become *structural* rules. In a Gentzen-style sequent calculus, formulating such structural rules would be impossible: one only has structural punctuation marks for $\otimes$ and $\oplus$ (the antecedent and succedent comma). But one could not formulate (20) as a *logical* rule either: it introduces two operations simultaneously.

$$\frac{X \cdot \otimes \cdot Y \vdash Z \cdot \oplus \cdot W}{Z \cdot \oslash \cdot X \vdash W \cdot / \cdot Y}\, G1 \qquad \frac{X \cdot \otimes \cdot Y \vdash Z \cdot \oplus \cdot W}{Y \cdot \oslash \cdot W \vdash X \cdot \setminus \cdot Z}\, G3$$
$$\frac{X \cdot \otimes \cdot Y \vdash Z \cdot \oplus \cdot W}{Z \cdot \oslash \cdot Y \vdash X \cdot \setminus \cdot W}\, G2 \qquad \frac{X \cdot \otimes \cdot Y \vdash Z \cdot \oplus \cdot W}{X \cdot \oslash \cdot W \vdash Z \cdot / \cdot Y}\, G4 \tag{24}$$

Logical rules For each connective there is a left and a right introduction rule. One of these is a one-premise rewrite rule, exchanging the logical connective for its structural counterpart; the other rule puts together a complex formula alongside the matching complex structure.

Rewrite rules $\$ \in \{\otimes, \oslash, \oslash\}$, $\# \in \{\oplus, \setminus, /\}$.

$$\frac{A \cdot \$ \cdot B \Rightarrow Y}{A \,\$\, B \Rightarrow Y}\, \$L \qquad \frac{X \Rightarrow A \cdot \# \cdot B}{X \Rightarrow A \,\#\, B}\, \#R \tag{25}$$

The rewrite rules are invertible. As an example, compare $(\otimes L)$ and $(\otimes L)^{-1}$.

$$\frac{A \cdot \otimes \cdot B \Rightarrow Y}{A \otimes B \Rightarrow Y}\, \otimes L \qquad \frac{\dfrac{A \Rightarrow A \quad B \Rightarrow B}{A \cdot \otimes \cdot B \Rightarrow A \otimes B}\, \otimes R \qquad A \otimes B \Rightarrow Y}{A \cdot \otimes \cdot B \Rightarrow Y}\, \mathrm{Cut} \tag{26}$$

Two premise rules The $(/L)$, $(\oslash R)$ rules are left-right symmetric.

$$\frac{X \Rightarrow A \quad Y \Rightarrow B}{X \cdot \otimes \cdot Y \Rightarrow A \otimes B}\, \otimes R \qquad \frac{A \Rightarrow X \quad B \Rightarrow Y}{A \oplus B \Rightarrow X \cdot \oplus \cdot Y}\, \oplus L \tag{27}$$

$$\frac{X \Rightarrow A \quad B \Rightarrow Y}{A \backslash B \Rightarrow X \cdot \setminus \cdot Y}\, \setminus L \qquad \frac{X \Rightarrow A \quad B \Rightarrow Y}{X \cdot \oslash \cdot Y \Rightarrow A \oslash B}\, \oslash R \tag{28}$$

Equivalence For every arrow $f : A \longrightarrow B$, there is a sequent proof $A \Rightarrow B$. For every sequent proof $X \Rightarrow Y$, there is an arrow $f : X^\circ \longrightarrow Y^\circ$, where X°, Y° are the formulas obtained from X, Y by replacing the structural connectives by their logical counterparts.

From arrows to sequent proofs 1_A and composition $g \circ f$ are immediate. We use the invertibility of the rewrite rules to prove the residuation/adjoints laws in the sequent calculus. Below, as an example, a sequent proof for $\triangleright f$.

$$\frac{f : A \otimes B \longrightarrow C}{\triangleright f : A \longrightarrow C/B} \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{A \otimes B \Rightarrow C}{A \cdot \otimes \cdot B \Rightarrow C}\,(\otimes L)^{-1}}{A \Rightarrow C \cdot / \cdot B}\,rp}{A \Rightarrow C/B}\,/R \tag{29}$$

From sequent proofs to arrows Under the mapping $\cdot^\circ$ Cut turns into composition of arrows, the (dual) display postulates into the (dual) residuation rules, and the distributivity postulates into the rule form of the arrows **d, q, b, p**, which in (21) we have shown to be derivable in **aLG**. For the logical group, the premise and conclusion of the rewrite rules are identified. The two-premise logical rules become the monotonicity rules — derivable rules of inference in **aLG** as we saw.

Cut Elimination, decidability (Moortgat 2007) In **sLG**, Cut is an admissible rule: every theorem has a cut-free derivation.

Decidability is a nice property to have. Yet, the astute reader at this point may feel disappointed: the goal-driven, backward-chaining, cut-free proof search of the decision procedure presupposes that the *structure* of the goal sequent is given. Parsing, as it is standardly understood, means deciding whether a string is well-formed, and assigning it a proper structure. Here, to start backward-chaining sequent proof search, we have to assume that the correct structure is already given. A generate-and-test approach, obviously, is not feasible here: the number of binary bracketings over a string of length n being the Catalan number C_n. We haven't addressed the parsing problem, in other words. Turning to *proof nets* in §2.2, this situation will change: the construction algorithm for **LG** nets will work in a *data-driven* mode, effectively *computing* the structure of the goal sequent.

2.2 Proof nets

Proof nets are a graphical way of representing proofs, introduced first for linear logic (Girard 1987). Proof nets can either be seen as a sort of "parallellized" sequent proofs or as a sort of multi-conclusion natural deduction. Proof nets are

defined as a subclass of a larger class of graphs called proof structures. Where proof nets correspond to sequent proofs, proof structures in general may not, but we can distinguish proof nets from other proof structures based only on properties of the graph.

The proof nets for the Lambek-Grishin calculus we present in this section are a simple extension of the proof nets for the multimodal Lambek calculus of (Moot and Puite 2002). A proof structure is a (hyper)graph where the vertices are labeled by formulas and the edges connect these formulas. In what follows we will often speak of formula occurrences (or simply *formulas* if there is no possibility of confusion) instead of vertices labeled by formulas. The hyperedges correspond to the logical rules, linking the active formulas and the main formula of the rule and keeping track of whether one is dealing with a non-invertible two-premise rule or with an invertible one-premise rule. We'll call these *tensor* and *cotensor* links respectively.

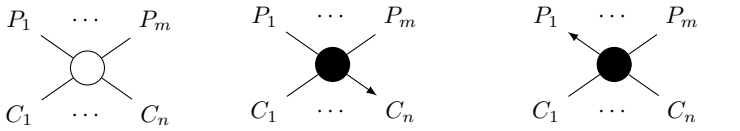
Proof structures and abstract proof structures

Definition 1. *A link is a tuple* $\langle t, p, c, m\rangle$ *where*

- t *is the type of the link — tensor or cotensor*
- p *is the list of premisses of the link,*
- c *is the list of conclusions of the link,*
- m*, the main vertex/formula of the link, is either a member of* p*, a member of* c *or the constant "nil".*

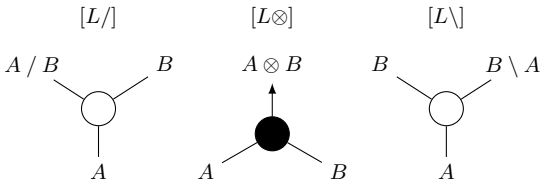
In case m *is a member of* p *we speak of a* left *link (corresponding to the left rules of the sequent calculus, where the main formula of the link occurs in the antecedent) and in case* m *is a member of* c *we speak of a* right *link.*

Graphically, links are displayed as shown below. A central node links together the premisses and conclusions of the link; when we need to refer to the connections between the central node and the vertices, we will call them its *tentacles*. The interior of this central node is white for a tensor link and black for a cotensor link. The premisses are drawn, in left-to-right order, above the central node and the conclusions, also in left-to-right order, are drawn below it. The main formula of cotensor links is drawn as an arrow to the member of the premisses or the conclusions which is the main formula of the link. The main formula of tensor links are not distinguished visually, but can be determined by inspection of the formula labels.

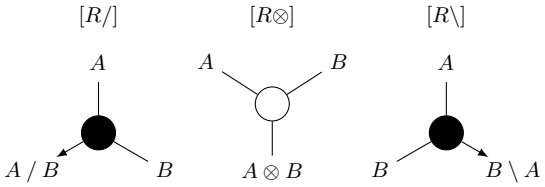


tensor rule cotensor rule (right rule) cotensor rule (left rule)

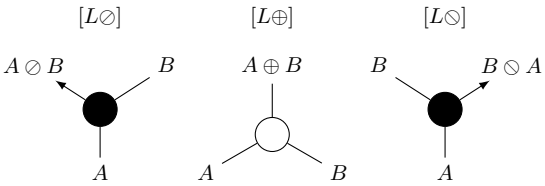
Lambek connectives — hypothesis



Lambek connectives — conclusion



Grishin connectives — hypothesis



Grishin connectives — conclusion

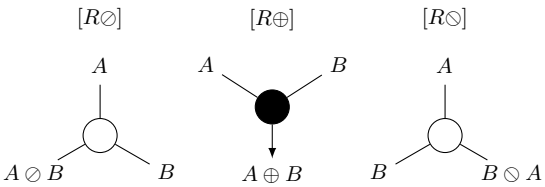


Fig. 1. Links for proof structures of the Lambek-Grishin calculus

Figure 1 shows the links for the Lambek-Grishin calculus: there are two links for each connective, one link where the main formula is a premiss of the link (a left link) and one link where the main formula is a conclusion of the link (a right link). The symmetry between the Lambek connectives and the Grishin connectives is immediately clear: the links for the Grishin connectives are up-down symmetric versions of the links for the Lambek connectives.

Definition 2. *A* proof structure $\langle S, \mathcal{L} \rangle$ *is a finite set of formula occurrences* S *and a set of links* $\mathcal{L}$ *from those shown in Figure 1 such that.*

- *each formula is at most once the premiss of a link,*
- *each formula is at most once the conclusion of a link.*

Formulas which are not the conclusion of any link are called the hypotheses *of the proof structure. Formulas which are not the premiss of any link are called the* conclusions *of the proof structure.*

We will say that a proof structure with hypotheses $H_1, \ldots, H_m$ *and conclusions* $C_1, \ldots, C_n$ *is a proof structure of* $H_1, \ldots, H_m \Rightarrow C_1, \ldots C_n$.

Example 1. Figure 2 shows the hypothesis unfolding of $(s \oslash s) \oslash np$ and the conclusion unfolding of $s \,/\, (np \setminus s)$. Both are obtained by simple application of the rules of Figure 1 until we reach the atomic subformulas.

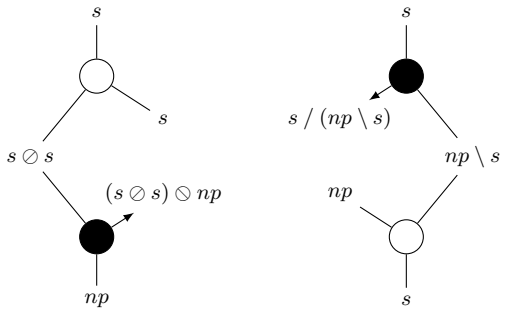


Fig. 2. Lexical unfolding

Though the figure satisfies the conditions of being a proof structure (note, for example, that connectedness is not a requirement, so a proof structure is allowed to have one connected component for each of the unfolded formulas), it is a proof structure of $(s \oslash s) \oslash np, s, s, np \Rightarrow s \,/\, (np \setminus s), s, s, np$. We can obtain a proof structure of $(s \oslash s) \oslash np \Rightarrow s \,/\, (np \setminus s)$ by identifying atomic formulas (this node identification corresponds to the "axiom links" of linear logic proof nets). In this case, we choose to identify the top s of the left subgraph with the bottom s

of the right subgraph and perform the unique choice for the remaining atomic formulas. The result is the proof structure shown in Figure 3 on the left.

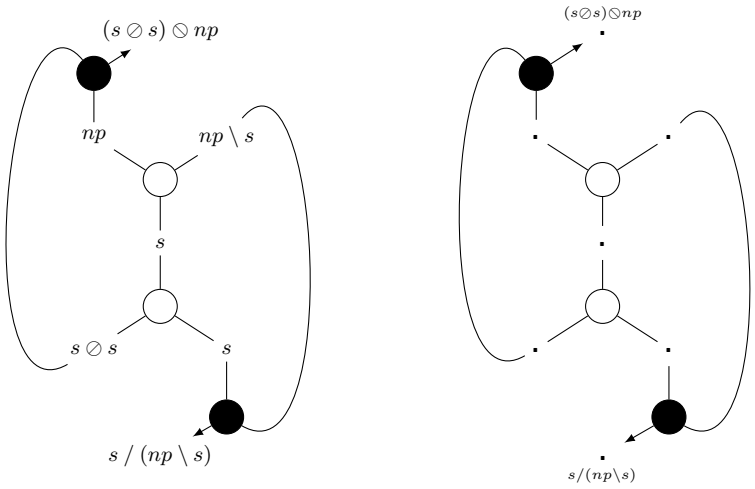


Fig. 3. Proof structure of $(s \oslash s) \oslash np \Rightarrow s \,/\, (np \setminus s)$ corresponding to the lexical unfolding of Figure 2 (left) and its corresponding abstract proof structure

Let's take a closer look at this new proof structure. We have conneced the minor premiss of the implication and co-implication links by a curve. This is due to the graphical constraints of writing these proof nets on the plane: we want to draw the $np \setminus s$ node *below* the cotensor link at the bottom of the figure, since it is a conclusion of this link, but would have to draw the figure on a cylinder to make this work — in other words, following down a path premiss - link - conclusion does not necessarily give a total order but can give a cyclic order on the formulas in the proof structure; for proof *nets*, these cyclic paths can only pass through the minor premiss of a cotensor (co-)implication link. As indicated by the drawing, the connection to the $np \setminus s$ node from the cotensor link arrives from above, indicating it is a conclusion of this link. Similarly, we go down from the $s \oslash s$ node to arrive at the other cotensor link.

A comparison with the introduction rule for the implication in natural deduction is another way to make this clear. For the introduction rule, we hypothesise a formula $np \setminus s$ (here, a conclusion of the cotensor rule), then derive s (here a premiss of the cotensor rule). The introduction rule then indicates we can withdraw this hypothesis and conclude $s \,/\, (np \setminus s)$, with some indexing indicating which hypotheses are withdrawn at which rule. In the proof structure above, the connection between the cotensor link and the $np \setminus s$ rule plays exactly the role of this indexing (though, since a proof structure is not necessarily a proof, we have

no guarantee yet that the introduction rule is correctly applied; the contractions introduced later will remedy this).

With this in mind, we can verify that the proof structure in Figure 3 corresponds exactly to the one in Figure 2 with the stated node identifications: we have the same formula occurrences and the links have the same premisses as well as the same conclusions.

So while the logical rules of a sequent proof correspond directly to the links of a proof net, the axioms and cut rules of a sequent proof correspond to *formulas*. An axiomatic formula is a formula which is not the main formula of any link. A cut formula is a formula which is the main formula of two links. So on the left of Figure 3, the *np* formula and both *s* formulas are axiomatic.

Definition 3. *An* abstract proof structure $\langle V, \mathcal{L}, h, c\rangle$ *is a set of vertices* V*, a set of (unlabeled) links* $\mathcal{L}$ *and two functions* h *and* c*, such that.*

- *each formula is at most once the premiss of a link,*
- *each formula is at most once the conclusion of a link,*
- h *is a function from the hypotheses of the abstract proof structure to formulas,*
- c *is a function from the conclusions of the abstract proof structure to formulas.*

Note that the abstract proof structure corresponding to a two formula sequent $A \Rightarrow B$ has only a single vertex v, with $h(v) = A$ and $c(v) = B$.

The transformation from proof structure to abstract proof structure is a forgetful mapping: we transform a proof structure into an abstract proof structure by erasing all formula information on the internal vertices, keeping only the formula labels of the hypotheses and the conclusions. Visually, we remove the formula labels of the graph and replace them by simple vertices (▪) and we indicate the results of the functions h and c above (resp. below) the vertices (those which are hypotheses and conclusions of the abstract proof structure respectively). As a result, we have to following four types of vertices in an abstract proof structure.

internal hypothesis conclusion both

Example 2. Figure 3 shows (on the right) the transformation of the proof structure on its left into an abstract proof structure. In the abstract proof structure, we can no longer distinguish which vertices are axioms: only the cotensor links still allow us to distinguish between the main and active vertices of the link by means of the arrow.

Definition 4. *A* tree *is an acyclic, connected abstract proof structure which does not contain any cotensor links.*

The trees of Definition 4 correspond to sequents in a rather direct way. In fact, they have the rather pleasant property of "compiling away" the display rules of the sequent calculus. Or, in other words, trees represent a class of sequents which is equivalent up to the display postulates.

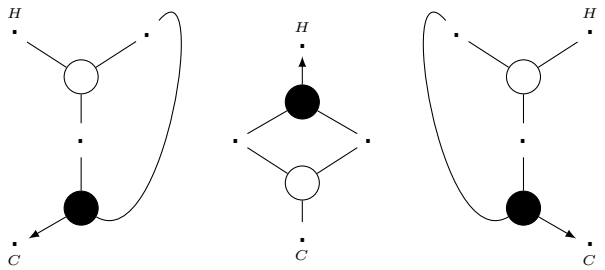


Fig. 4. Contractions — Lambek connectives

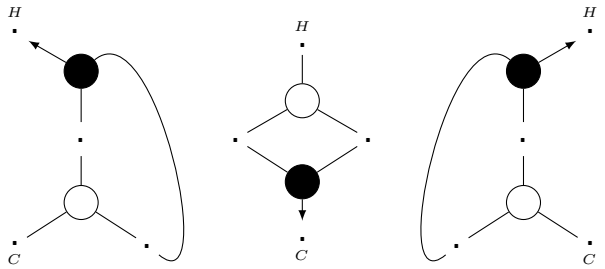


Fig. 5. Contractions — Grishin connectives

Definition 5. *Given an abstract proof structure A, we say that A* contracts in one step *to A', written $A \to A'$ iff A' is obtained from A by replacing one of the subgraphs of the form shown in Figures 4 and 5 by a single vertex.*

H represents the result of the function h for the indicated node (relevant only in case this node is a hypothesis of the abstract proof structure). Similarly, C represent the formula assigned by the function c to the indicated node.

Given an abstract proof structure A we say that A contracts to *an abstract proof structure A′ if there is a sequence of zero or more one step contractions from A to A′.*

When we say that a proof structure *P contracts to an abstract proof structure A′ we will mean that the underlying abstract proof structure A of P contracts to A′.*

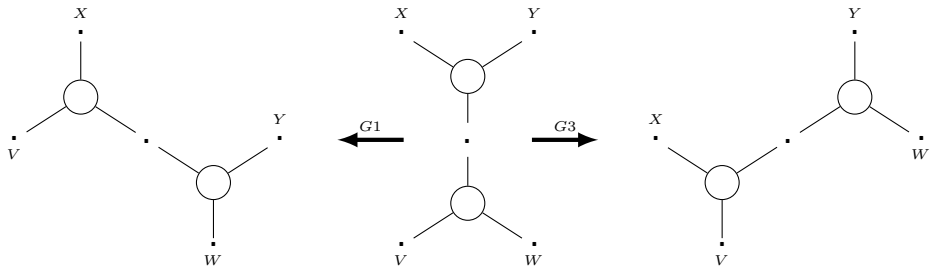


Fig. 6. Grishin interactions I — "mixed associativity"

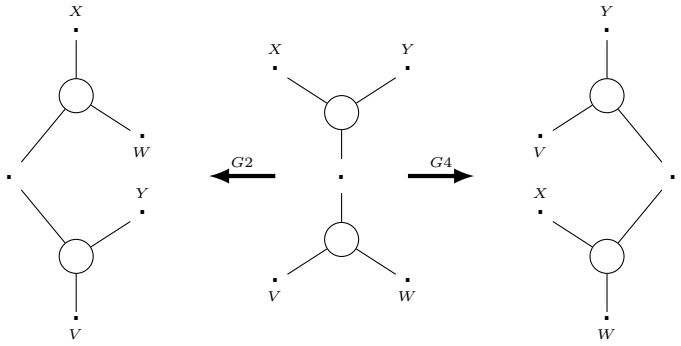


Fig. 7. Grishin interactions II — "mixed commutativity"

As we saw in §1.2, to obtain expressivity beyond context-free, we are interested in **LG** with added interaction principles. The (rule forms of the) postulates (§18) correspond to additional rewrite rules on the abstract proof structures. Figures 6 and 7 give the rewrite rules corresponding to the postulates **d**, **b** and **q**, **p** respectively[3]; a total of four rewrite rules ($G1$) to ($G4$). All four rewrite

[3] These are Grishin's Class IV interactions. His Class I can be obtained by inversing all four arrows in the two figures.

rules start from the same inital configuration and replace it by one of the four possible configurations indicated in the figures.

Proof nets

Definition 6. *A proof structure P is a* proof net *iff its underlying abstract proof structure A converts to a tree using the contractions of Figures 4 and 5 and the structural rules of Figures 6 and 7.*

Example 3. To show that the proof structure of Figure 3 is a proof net, we need to show it can be contracted to a tree. Inspection of the contractions shows that none of them apply, but the interaction rules do: the two tensor links in the center of the figure are in the right configuration for the interaction rules. Applying rule $(G1)$ produces the abstract proof structure shown in Figure 8 on the right.

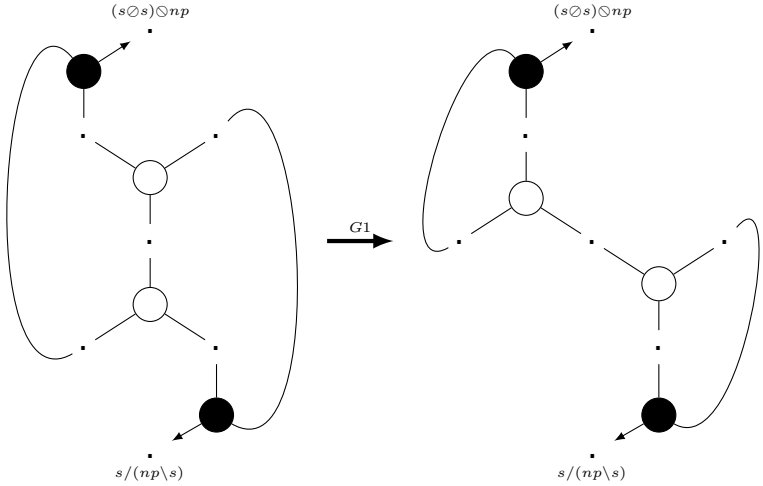


Fig. 8. Applying rule $(G1)$ to the abstract proof structure of Figure 3

Now, we are in the right structure to contract the two cotensor links. Any order is possible. Figure 9 shows the result of first applying the $(L\otimes)$, then the $(R/)$ contraction.

Example 4. Figure 10 shows the lexical proof structures for a generalized quantifier noun phrase, a transitive verb, a determiner and a lexical noun.

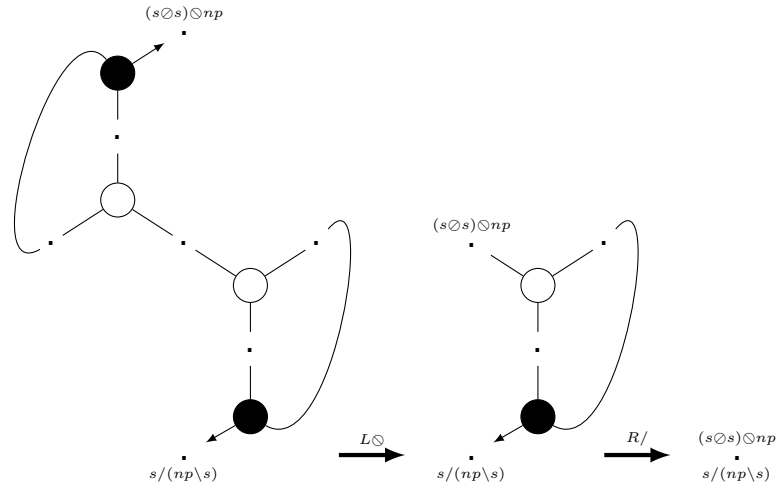


Fig. 9. Applying rule ($L\oslash$) and ($R/$) contractions to the abstract proof structure of Figure 8

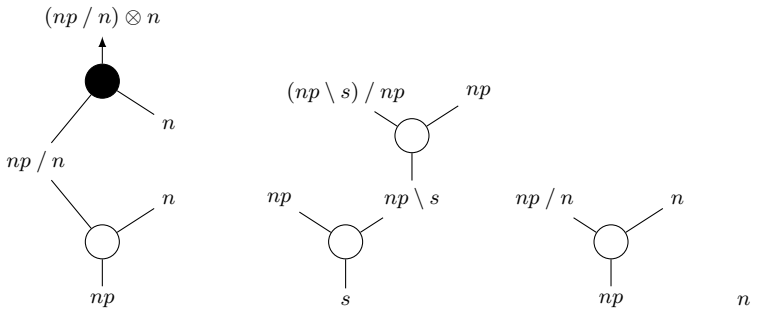


Fig. 10. Lexical proof structures for a generalized quantifier noun phrase, a transitive verb, a determiner and a noun

Figure 11 gives, on the left, one of several possible identifications of n and np formulas, but the only one which produces a proof net with the lexical entries in the indicate order and the corresponding abstract proof structure on the right. This abstract proof structure allows us to apply a contraction directly, as shown in Figure 12.

Theorem 1. *A proof structure P is a proof net — that is, P converts to a tree T — iff there is a sequent proof of T.*

The proof is an easy adaptation of the proof of (Moot and Puite 2002). A detailed proof can be found in (Moot 2007).

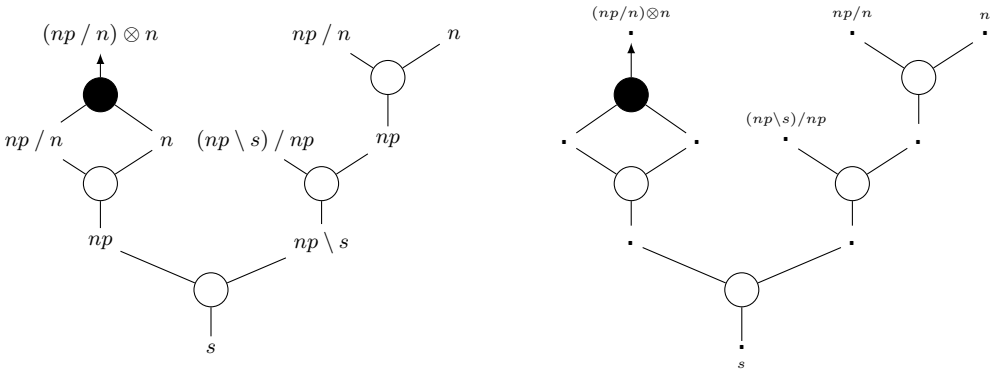


Fig. 11. Judgement $(np/n)\otimes n, (np\backslash s)/np, np/n, n \Rightarrow s$: proof structure and abstact proof structure

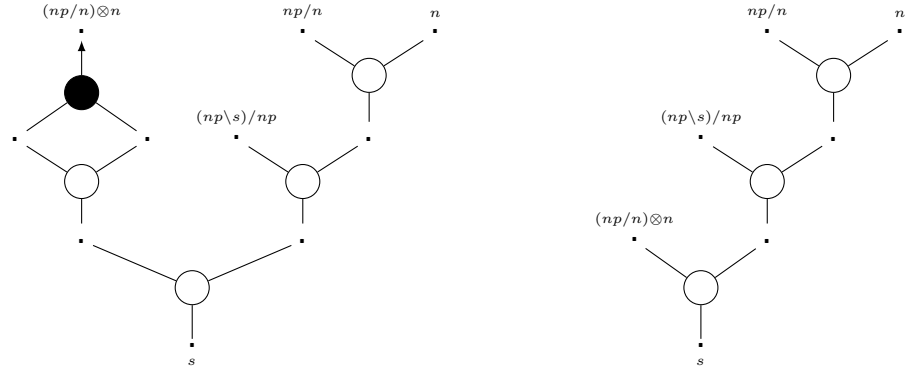


Fig. 12. Abstract proof structure and contraction

By requiring a proof structure to contract to a tree, we actually compute the structure of the antecedent, which is a pleasant property.

We will look a bit more at the structure of the conversion sequence in what follows and the following definition will be useful in this context.

Definition 7. *Given a proof net P, a* component *C of P is a maximal subnet of P containing only tensor links.*

From a proof net, we can obtain its components by simply erasing all cotensor links. The components will be the connected components (in the graph-theoretic sense) of the resulting graph. In what follows we will implicitly use the word component to refer only to components containing at least one tensor link. Though there is no problem in allowing a component to be a single vertex, the correspondence between focused sequent proofs and proof nets is more clear when components are non-trivial.

Generalized contractions As can been seen from the figures, the interaction rule introduce nondeterminism in proof search: a single subtree can be rewritten in four different trees and this applies recursively for the depth of a component. However, this is not as bad as it seems: in many cases, we can "compile away" the interaction principles by permitting contractions in a larger set of configurations than those shown in Figures 4 and 5. The contractions for the product and co-product stay the same, but the contractions for the implications and co-implications will change as shown in Figures 13 and 14. In Figure 13, the contraction can apply iff there is a path of Grishin tensor links connecting the two portrayed points above and below the substructure in the figure. In case this path is empty, the normal contraction applies and in case this path has length greater than one, then, by construction, the Lambek tensor link is connected to a Grishin tensor link, and there is a path from this link through the displayed substructure. If this path goes left from the first link, we can apply rule $(G2)$ and reduce the distance. If this path goes right from this first Grishin link, we can apply rule $(G1)$ and reduce the distance as well — in the case of the $R/$ contraction — or $(G3)$ and $(G4)$ — in the case of the $R\backslash$ contraction.

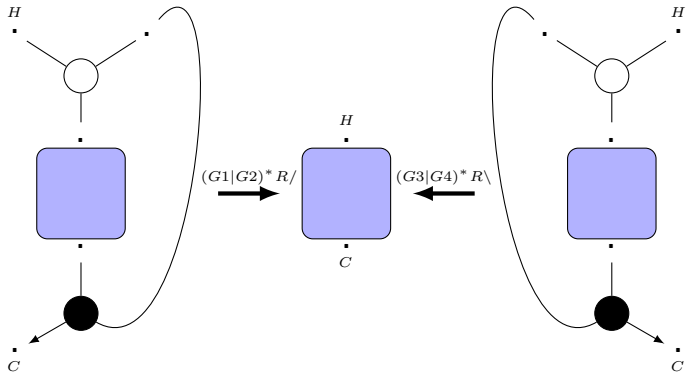


Fig. 13. Derived Contractions — Lambek

By up-down symmetry, the contractions of Figure 14 require a path of Lambek tensor connectives with the interaction principles listed. Note that it suffices to compute one case: the other cases follow from up-down symmetry and left-to-right symmetry between the interaction principles and the contractions.

The derived contractions allow us to simplify the reduction sequences considerably. It is even the case that, whenever the result tree contains only a single type of constructors (that is, only Grishin tensor links or only Lambek tensor

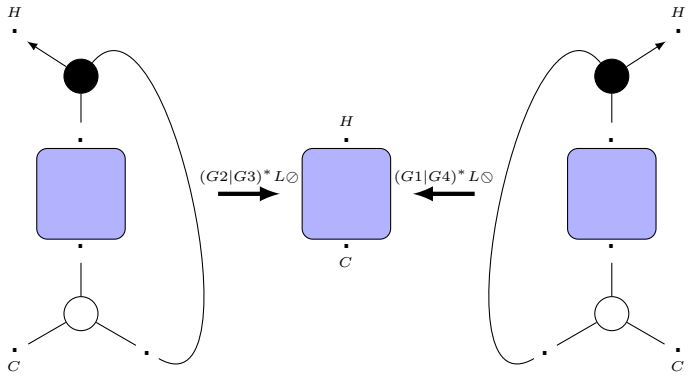


Fig. 14. Derived Contractions — Grishin

links) then we can replace the interaction principles by the generalized contractions.

Summary: Proof nets and sequent proofs As a useful summary of the correspondence between proof nets and sequent proofs, we give the following table.

sequent calculus	proof structure	conversion
axiom	axiomatic formula	—
cut	cut formula	—
two-premise rule	tensor link	—
one-premise rule	cotensor link	contraction
interaction rule	—	rewrite

The invertible one-premise rules correspond to both a link and a contraction and the interaction rules are invisible in the proof structure, appearing only in the conversion sequence.

With a bit of extra effort in the sequentialization proof — and the exclusion of cuts on axioms, because like natural deduction, we cannot distinguish between the following two sequent proofs

$$\dfrac{\dfrac{}{A \Rightarrow A}\ Ax \quad \dfrac{}{A \Rightarrow A}\ Ax}{A \Rightarrow A}\ Cut \qquad\qquad \dfrac{}{A \Rightarrow A}\ Ax$$

— we can show that these correspondences are 1-on-1, that is each axiomatic formula in a proof net corresponds to exactly one axiom rule in the sequent proof, each non-invertible two-premise rule corresponds to exactly one link in

the proof net and each invertible one-premise rule to exactly one link in the proof net and exactly one contraction in its conversion sequence.

Discussion Proof nets provide a solution to the spurious ambiguity problem of sequent calculus proof search: because of inessential, bureaucratic rule permutations we can have multiple sequent calculus proofs for what, in essence, corresponds to the same proof (which corresponds semantically to a different *reading* of the phrase under consideration). Proof nets, like (product-free) natural deduction, have different proof objects only for proofs of a judgement which differ essentially. In addition, the combinatorial possibilities for such readings, which are obtained by finding a complete matching of the premiss and conclusions atomic formulas, can easily be enumerated for a given sequence of formulas.

So proof nets have a 1-1 correspondence between proofs and readings, compute the structure of the sequents, give a graphical representation which makes the display postulates superfluous and, in certain cases, can hide the interaction rules by using generalized contractions.

3 Proof nets and focused display calculus

The spurious non-determinism of naive backward-chaining proof search can be also addressed within the sequent calculus itself, by introducing an appropriate notion of 'normal' derivations. In §3.1, we introduce **fLG**, a focused version of the sequent calculus for **LG**. In §3.2, we then study how to interpret focused derivations from a proof net perspective.

3.1 fLG: focused display calculus

The strategy of focusing has been well-studied in the context of linear logic, starting with the work of Andreoli (Andreoli 2001). It is based on the distinction between *asynchronous* and *synchronous* non-atomic formulas. The introduction rule for the main connective of an asynchronous formula is *invertible*; it is non-invertible for the synchronous formulas. Backward chaining focused proof search starts with an asynchronous phase where invertible rules are applied deterministically until no more candidate formulas remain. At that point, a non-deterministic choice for a synchronous formula must be made: this formula is put 'in focus', and decomposed in its subformulae by means of non-invertible rules until no more non-invertible rules are applicable, at which point one reenters an asynchronous phase. The main result of (Andreoli 2001) is that focused proofs are complete for linear logic.

Focused proof search for the Lambek-Grishin calculus has been studied by Bastenhof (2011) who uses a one-sided presentation of the calculus. In this section, we implement his focusing regime in the context of the two-sided sequent format of (Bernardi and Moortgat 2010). We proceed in two steps. First we introduce **fLG**, the focused version of the sequent calculus of §2.1. **fLG** makes a distinction between focused and unfocused judgements, and has a set of inference rules to switch between these two. **fLG** comes with a term language that is in Curry-Howard correspondence with its derivations. This term language is a directional refinement of the $\overline{\lambda}\mu\widetilde{\mu}$ language of (Curien and Herbelin 2000).

The second step is to give a constructive interpretation for **LG** derivations by means of a continuation-passing-style translation: a mapping $\lceil\cdot\rceil$ that sends derivations of the multiple-conclusion source logic to (natural deduction) proofs in a fragment of single-conclusion intuitionistic Linear Logic MILL (in the categorial terminology: **LP**). For the translation of (Bastenhof 2011) that we follow here, the target fragment has linear products and negation $A^{\perp}$, i.e. a restricted form of linear implication $A \multimap \perp$, where $\perp$ is a distinguished atomic type, the response type. Focused source derivations then can be shown to correspond to distinct *normal* natural deduction proofs in the target calculus.

$$\mathbf{fLG}^{\mathcal{A}}_{/,\otimes,\backslash,\oslash,\oplus,\otimes} \xrightarrow{\lceil\cdot\rceil} \mathbf{LP}^{\mathcal{A}\cup\{\perp\}}_{\otimes,\cdot^{\perp}} \quad \left(\xrightarrow{\cdot^{\ell}} \mathbf{IL}^{\{e,t\}}_{\times,\to}\right)$$

For the linguistic illustrations in §3.2, we compose the CPS translation $\lceil\cdot\rceil$ with a second mapping $\cdot^{\ell}$, that establishes the connection with Montague-style semantic representations. This mapping sends the linear constructs to their intuitionistic counterparts, and allows *non-linear* meaning recipes for the translation of the lexical constants.

fLG: proofs and terms We set up **fLG** in the Curry-Howard proofs-as-programs fashion, starting from a term language for which the sequent logic then provides the type system. The term language encodes the *logical* steps of a derivation (left and right introduction rules, and the new set of left and right (de)focusing rules, to be introduced below); structural rules (residuation, distributivity) leave no trace in the proof terms.

Sequent structures, as in §2.1, are built out of formulas. Input formulas now are labeled with variables $x, y, z, \ldots$, output formulas with covariables $\alpha, \beta, \gamma, \ldots$. To implement the focusing regime, we allow sequents to have one displayed formula *in focus*. Writing the focused formula in a box, **fLG** will have to deal with three types of judgements: sequents with no formula in focus (we'll call these *structural*), and sequents with a succedent or antecedent formula in focus.

$$X \vdash Y \qquad X \vdash \boxed{A} \qquad \boxed{A} \vdash Y$$

Corresponding to the types of sequents, the term language has three types of expressions: commands, values and contexts respectively. For commands, we use the metavariables c, C, for values v, V, for contexts e, E. The typing rules below provide the motivation for the subclassification.

$$\begin{array}{c} v ::= \mu\alpha.C \mid V \quad ; \quad V ::= x \mid v_1 \otimes v_2 \mid v \oslash e \mid e \odot v \\ e ::= \widetilde{\mu}x.C \mid E \quad ; \quad E ::= \alpha \mid e_1 \oplus e_2 \mid v \backslash e \mid e / v \\ c ::= \langle x \upharpoonright E \rangle \mid \langle V \upharpoonright \alpha \rangle \\ C ::= c \mid \tfrac{x\ y}{z}.C \mid \tfrac{x\ \beta}{z}.C \mid \tfrac{\beta\ x}{z}.C \mid \tfrac{\alpha\ \beta}{\gamma}.C \mid \tfrac{x\ \beta}{\gamma}.C \mid \tfrac{\beta\ x}{\gamma}.C \end{array} \tag{30}$$

Typing rules To enforce the alternation between asynchronous and synchronous phases of focused proof search, formulas are associated with a polarity: *positive* for non-atomic formulas with invertible left introduction rule: $A \otimes B$, $A \oslash B$, $B \odot A$; *negative* for non-atomic formulas with invertible right introduction rule: $A \oplus B$, $A \backslash B$, B / A. For atomic formulas, one can fix an arbitrary polarity. Different choices lead to different prooftheoretic behaviour (and to different interpretations, once we turn to the CPS translation). We will assume that atoms are assigned a bias (positive or negative) in the lexicon. Below the typing rules for **fLG** (restricting attention to the cut-free system).

(Co-)Axiom, (de)focusing

$$\frac{}{x : A \vdash \boxed{x : A}}\ \mathrm{Ax} \qquad \frac{}{\boxed{\alpha : A} \vdash \alpha : A}\ \mathrm{CoAx}$$

$$\frac{X \vdash \boxed{V : A}}{\langle V \upharpoonright \alpha \rangle : (X \vdash \alpha : A)}\ \mu^* \qquad \frac{\boxed{E : A} \vdash X}{\langle x \upharpoonright E \rangle : (x : A \vdash X)}\ \widetilde{\mu}^*$$

$$\frac{C : (x : A \vdash X)}{\boxed{\widetilde{\mu}x.C : A} \vdash X}\ \widetilde{\mu} \qquad \frac{C : (X \vdash \alpha : A)}{X \vdash \boxed{\mu\alpha.C : A}}\ \mu$$

First we have the focused version of the axiomatic sequents, and rules for focusing and defocusing which are new with respect to the unfocused presentation of §2.1. There is a polarity restriction on the formula A in these rules: the boxed formula has to be negative for $\mathrm{CoAx}, \mu, \widetilde{\mu}^*$; for $\mathrm{Ax}, \widetilde{\mu}, \mu^*$ it has to be positive. In the (Co-)Axiom cases, A can be required to be atomic.

From a backward-chaining perspective, the $\mu, \widetilde{\mu}$ rules *remove* the focus from a focused succedent or antecedent formula. The result is an unfocused premise

sequent, the domain of applicability of the invertible rules, i.e. one enters the asynchronous phase. From the same perspective, the rules $\mu^*, \widetilde{\mu}^*$ place a succedent or antecedent formula in focus, shifting control to the non-invertible rules of the synchronous phase. The $\mu^*, \widetilde{\mu}^*$ rules are in fact instances of Cut where one of the premises is axiomatic.

Invertible rules The term language makes a distinction between simple commands c (the image of the focusing rules $\widetilde{\mu}^*, \mu^*$: $\langle x \upharpoonright E\rangle, \langle V \upharpoonright \alpha\rangle$) from extended commands C. The latter start with a sequence of invertible rewrite rules replacing a logical connective by its structural counterpart. We impose the requirement that in the asynchronous phase all formulas to which an invertible rule is applicable are indeed decomposed.

$$\frac{C : (x : A \cdot \otimes \cdot y : B \vdash X)}{\frac{x\ y}{z}.C : (z : A \otimes B \vdash X)}\otimes L \qquad \frac{C : (X \vdash \alpha : A \cdot \oplus \cdot \beta : B)}{\frac{\alpha\ \beta}{\gamma}.C : (X \vdash \gamma : A \oplus B)}\oplus R$$

$$\frac{C : (x : A \cdot \oslash \cdot \beta : B \vdash X)}{\frac{x\ \beta}{z}.C : (z : A \oslash B \vdash X)}\oslash L \qquad \frac{C : (X \vdash x : A \cdot \backslash \cdot \beta : B)}{\frac{x\ \beta}{\gamma}.C : (X \vdash \gamma : A\backslash B)}\backslash R$$

$$\frac{C : (\beta : B \cdot \oslash \cdot x : A \vdash X)}{\frac{\beta\ x}{z}.C : (z : B \oslash A \vdash X)}\oslash L \qquad \frac{C : (X \vdash \beta : B \cdot / \cdot x : A)}{\frac{\beta\ x}{\gamma}.C : (X \vdash \gamma : B/A)}/R$$

Non-invertible rules When a positive (negative) formula has been brought into focus in the succedent (antecedent), one is committed to transfer the focus to its subformulae.

$$\frac{\boxed{e_1 : B} \vdash Y \qquad \boxed{e_2 : A} \vdash X}{\boxed{e_1 \oplus e_2 : B \oplus A} \vdash Y \cdot \oplus \cdot X}\oplus L \qquad \frac{X \vdash \boxed{v_1 : A} \qquad Y \vdash \boxed{v_2 : B}}{X \cdot \otimes \cdot Y \vdash \boxed{v_1 \otimes v_2 : A \otimes B}}\otimes R$$

$$\frac{X \vdash \boxed{v : A} \qquad \boxed{e : B} \vdash Y}{\boxed{v\backslash e : A\backslash B} \vdash X \cdot \backslash \cdot Y}\backslash L \qquad \frac{X \vdash \boxed{v : A} \qquad \boxed{e : B} \vdash Y}{X \cdot \oslash \cdot Y \vdash \boxed{v \oslash e : A \oslash B}}\oslash R$$

$$\frac{\boxed{e : B} \vdash Y \qquad X \vdash \boxed{v : A}}{\boxed{e/v : B/A} \vdash Y \cdot / \cdot X}/L \qquad \frac{\boxed{e : B} \vdash Y \qquad X \vdash \boxed{v : A}}{Y \cdot \oslash \cdot X \vdash \boxed{e \oslash v : B \oslash A}}\oslash R$$

Derived inference rules: focus shifting To highlight the correspondence with the algorithm for proof net construction to be discussed in §2.2, we will use a derived rule format for shifting between a conclusion and premise focused formula. A branch from $(\widetilde{\mu}^*)$ via a sequence (possibly empty) of structural rules and rewrite rules to (μ) is compiled in a derived inference rule with the $\widetilde{\mu}^*$ restrictions on A and the μ restrictions on B.

$$\dfrac{\dfrac{\boxed{E : A} \vdash Y}{\langle x \restriction E\rangle : (x : A \vdash Y)}\ \widetilde{\mu}^*}{\begin{array}{c}\vdots\\ (\mathit{res}, \mathit{distr}, \mathit{rewrite})\\ \vdots\\ (\div)\langle x \restriction E\rangle : (X \vdash \beta : B)\end{array}}\ \mu \qquad \rightsquigarrow \qquad \dfrac{\boxed{E : A} \vdash Y}{X \vdash \boxed{\mu\beta.(\div)\langle x \restriction E\rangle : B}}\ \leftrightharpoons$$

For the combinations of $\mu^*, \widetilde{\mu}^*$ and $\mu, \widetilde{\mu}$, this results in the focus shifting rules below. We leave it to the reader to add the terms.

$$\dfrac{\boxed{A} \vdash Y}{X \vdash \boxed{B}}\ \leftrightharpoons \qquad \dfrac{X' \vdash \boxed{A}}{X \vdash \boxed{B}}\ \Rightarrow \qquad \dfrac{X \vdash \boxed{A}}{\boxed{B} \vdash Y}\ \rightleftharpoons \qquad \dfrac{\boxed{A} \vdash Y'}{\boxed{B} \vdash Y}\ \Leftarrow \tag{31}$$

Illustrations We illustrate the effect of the focusing regime with some alternative ways of assigning a polarity bias to atomic formulas with a simple Subject-Transitive Verb-Object sentence. Examples with lexical material filled in would be 'everyone seeks/finds a unicorn'.

$$(np/n \otimes n) \cdot \otimes \cdot ((np\backslash s)/np \cdot \otimes \cdot (np/n \cdot \otimes \cdot n)) \vdash s \tag{32}$$

For the Object we have a Determiner-Noun combination. For the Subject, we take a product type $(np/n) \otimes n$, so that we have a chance to illustrate the working of the asynchronous phase of the derivation. In the unfocused sequent calculus **sLG**, this sequent has at least seven proofs, depending on the order of application of the introduction rules for the five occurrences of the logical connectives involved: $\otimes$ (once), / (three times), $\backslash$ (once).

What about the focused calculus **fLG**? Before answering this question, we have to decide on the polarization of the atomic types. Suppose we give them uniform negative bias. There is only one focused proof then: 'goal driven', top-down, to use parsing terminology. In the proof terms, we write tv for the transitive verb;

det for the object determiner; noun for the object common noun; subj for the subject noun phrase.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\boxed{n}^{\gamma'} \vdash n}{\boxed{np}^{\gamma} \vdash np \quad n \vdash \boxed{n}} \leftrightarrows}{\boxed{np/n} \vdash np \cdot / \cdot n} /L}{np/n \cdot \otimes \cdot n \vdash \boxed{np}} \leftrightarrows \quad \boxed{s}^{\beta} \vdash s}{\boxed{np\backslash s} \vdash (np/n \cdot \otimes \cdot n) \cdot \backslash \cdot s} \backslash L \quad \dfrac{\dfrac{\boxed{n}^{\alpha'} \vdash n}{\boxed{np}^{\alpha} \vdash np \quad n \vdash \boxed{n}} \leftrightarrows}{\boxed{np/n} \vdash np \cdot / \cdot n} /L}{np/n \cdot \otimes \cdot n \vdash \boxed{np}} \leftrightarrows}{\boxed{(np\backslash s)/np} \vdash ((np/n \cdot \otimes \cdot n) \cdot \backslash \cdot s) \cdot / \cdot (np/n \cdot \otimes \cdot n)} /L}{(np/n) \otimes n \cdot \otimes \cdot ((np\backslash s)/np \cdot \otimes \cdot (np/n \cdot \otimes \cdot n)) \vdash \boxed{s}} \leftrightarrows$$

$$\mu\beta.(\frac{y\ z}{\mathsf{subj}}.\langle\ \mathsf{tv} \uparrow ((Q \setminus \beta) \,/\, Q')\ \rangle) \quad \text{with}$$
$$Q : \mu\gamma.\langle\ y \uparrow (\gamma \,/\, \mu\gamma'.\langle\ z \uparrow \gamma'\rangle)\rangle\ ,\ Q' : \mu\alpha.\langle\ \mathsf{det} \uparrow (\alpha \,/\, \mu\alpha'.\langle\ \mathsf{noun} \uparrow \alpha'\rangle)\rangle \tag{33}$$

As an alternative, suppose basic type s keeps its negative bias, resetting the sentence continuation for each clausal domain, but the other basic types are assigned positive bias. We now have *two* focused derivations: 'data driven', bottom-up. To make sense of this difference, we will have to look at the CPS translation of these proofs, to be introduced below.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{np \overset{x_1}{\vdash} \boxed{np} \quad \boxed{s^-}^{\alpha_0} \vdash s^-}{\boxed{np\backslash s^-} \vdash np \cdot \backslash \cdot s^-} \backslash L \quad np \overset{y_1}{\vdash} \boxed{np}}{\boxed{(np\backslash s^-)/np} \vdash (np \cdot \backslash \cdot s^-) \cdot / \cdot np} /L}{\boxed{np} \vdash (np\backslash s^-)/np \cdot \backslash \cdot (np \cdot \backslash \cdot s^-)} \leftrightarrows \quad n \overset{\mathsf{noun}}{\vdash} \boxed{n}}{\boxed{np/n} \vdash ((np\backslash s^-)/np \cdot \backslash \cdot (np \cdot \backslash \cdot s^-)) \cdot / \cdot n} /L}{\boxed{np} \vdash s^- \cdot / \cdot ((np\backslash s^-)/np \cdot \otimes \cdot (np/n \cdot \otimes \cdot n))} \leftrightarrows \quad n \overset{z_0}{\vdash} \boxed{n}}{\boxed{np/n} \vdash (s^- \cdot / \cdot ((np\backslash s^-)/np \cdot \otimes \cdot (np/n \cdot \otimes \cdot n))) \cdot / \cdot n} /L}{(np/n) \otimes n \cdot \otimes \cdot ((np\backslash s^-)/np \cdot \otimes \cdot (np/n \cdot \otimes \cdot n)) \vdash \boxed{s^-}} \leftrightarrows$$

$$\mu\alpha.(\frac{x'\ z}{\mathsf{subj}}.\langle\ x'\ \rceil\ (\widetilde{\mu}x.\langle\ \mathsf{det}\ \rceil\ (\widetilde{\mu}y.\langle\ \mathsf{tv}\ \rceil\ ((x\setminus\alpha)\ /\ y)\ \rangle\ /\ \mathsf{noun})\ \rangle\ /\ z)\ \rangle) \qquad (34)$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{np \overset{x}{\vdash} \boxed{np} \quad \boxed{s^-} \overset{\alpha}{\vdash} s^-}{\boxed{np\backslash s^-} \vdash np\cdot\backslash\cdot s^-}\backslash L \qquad np \overset{y}{\vdash} \boxed{np}}{\boxed{(np\backslash s^-)/np} \vdash (np\cdot\backslash\cdot s^-)\cdot/\cdot np}/L}{\boxed{np} \vdash s^-\cdot/\cdot((np\backslash s^-)/np\cdot\otimes\cdot np)} \Leftarrow \qquad n \overset{z}{\vdash} \boxed{n}}{\boxed{np/n} \vdash (s^-\cdot/\cdot((np\backslash s^-)/np\cdot\otimes\cdot np))\cdot/\cdot n}/L}{\boxed{np} \vdash (np\backslash s^-)/np\cdot\backslash\cdot((np/n\cdot\otimes\cdot n)\cdot\backslash\cdot s^-)} \Leftarrow \qquad n \overset{\mathsf{noun}}{\vdash} \boxed{n}}{\boxed{np/n} \vdash ((np\backslash s^-)/np\cdot\backslash\cdot((np/n\cdot\otimes\cdot n)\cdot\backslash\cdot s^-))\cdot/\cdot n}/L}{(np/n)\otimes n\cdot\otimes\cdot((np\backslash s^-)/np\cdot\otimes\cdot(np/n\cdot\otimes\cdot n)) \vdash \boxed{s^-}} \Leftrightarrow$$

$$\mu\alpha.(\frac{x'\ z}{\mathsf{subj}}.\langle\ \mathsf{det}\ \rceil\ (\widetilde{\mu}y.\langle\ x'\ \rceil\ (\widetilde{\mu}x.\langle\ \mathsf{tv}\ \rceil\ ((x\setminus\alpha)\ /\ y)\ \rangle\ /\ z)\ \rangle\ /\ \mathsf{noun})\ \rangle) \qquad (35)$$

CPS translation Let us turn then to the translation that associates the proofs of the multiple-conclusion source logic **fLG** with a constructive interpretation, i.e. a linear lambda term of the target logic MILL/**LP**. CPS translations for **LG** were introduced in (Bernardi and Moortgat 2007; Bernardi and Moortgat 2010), who adapt the call-by-value and call-by-name regimes of (Curien and Herbelin 2000) to a directional environment. The translation of (Bastenhof 2011) (following (Girard 1991)) is an improvement in that it avoids the 'administrative redexes' of the earlier approaches: the image of **LG** source derivations, under the mapping from (Bastenhof 2011) that we present below, are *normal* **LP** terms.

The target language, on the type level, has the same atoms as the source language, and in addition a distinguished atom $\perp$, the response type. Complex types are linear products $-\otimes-$ and a defined negation $A^\perp \doteq A \multimap \perp$. The CPS translation $\lceil\cdot\rceil$ maps **fLG** source types, sequents and their proof terms to the target types and terms in Curry-Howard correspondence with normal natural deduction proofs.

Types For positive atoms, $\lceil p\rceil = p$, for negative atoms $\lceil p\rceil = p^\perp$. For complex types, the value of $\lceil\cdot\rceil$ depends on the polarities of the subtypes as shown in Table 1.

Table 1. CPS translation: non-atomic types

$\mathsf{pol}(A)$	$\mathsf{pol}(B)$	$\lceil A \otimes B \rceil$	$\lceil A/B \rceil$	$\lceil B \backslash A \rceil$
$-$	$-$	$\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp$	$\lceil A \rceil \otimes \lceil B \rceil^\perp$	$\lceil B \rceil^\perp \otimes \lceil A \rceil$
$-$	$+$	$\lceil A \rceil^\perp \otimes \lceil B \rceil$	$\lceil A \rceil \otimes \lceil B \rceil$	$\lceil B \rceil \otimes \lceil A \rceil$
$+$	$-$	$\lceil A \rceil \otimes \lceil B \rceil^\perp$	$\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp$	$\lceil B \rceil^\perp \otimes \lceil A \rceil^\perp$
$+$	$+$	$\lceil A \rceil \otimes \lceil B \rceil$	$\lceil A \rceil^\perp \otimes \lceil B \rceil$	$\lceil B \rceil \otimes \lceil A \rceil^\perp$

$\mathsf{pol}(A)$	$\mathsf{pol}(B)$	$\lceil A \oplus B \rceil$	$\lceil A \oslash B \rceil$	$\lceil B \oslash A \rceil$
$-$	$-$	$\lceil A \rceil \otimes \lceil B \rceil$	$\lceil A \rceil^\perp \otimes \lceil B \rceil$	$\lceil B \rceil \otimes \lceil A \rceil^\perp$
$-$	$+$	$\lceil A \rceil \otimes \lceil B \rceil^\perp$	$\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp$	$\lceil B \rceil^\perp \otimes \lceil A \rceil^\perp$
$+$	$-$	$\lceil A \rceil^\perp \otimes \lceil B \rceil$	$\lceil A \rceil \otimes \lceil B \rceil$	$\lceil B \rceil \otimes \lceil A \rceil$
$+$	$+$	$\lceil A \rceil^\perp \otimes \lceil B \rceil^\perp$	$\lceil A \rceil \otimes \lceil B \rceil^\perp$	$\lceil B \rceil^\perp \otimes \lceil A \rceil$

Terms The action of $\lceil \cdot \rceil$ on terms is given in (36). We write $\widetilde{x}, \widetilde{\alpha}$ for the target variables corresponding to source x, α. The (de)focusing rules correspond to application/abstraction in the target language. Non-invertible (two premise) rules are mapped to linear pair terms; invertible rewrite rules to the matching deconstructor, the **case** construct (ϕ, ψ, ξ metavariables for the the (co)variables involved).

$$\begin{array}{rc} \textit{(co)var} & \lceil x \rceil = \widetilde{x} \quad ; \quad \lceil \alpha \rceil = \widetilde{\alpha} \\ \textit{linear application} & \lceil \langle x \upharpoonright E \rangle \rceil = (\widetilde{x}\ \lceil E \rceil)\ ; \quad \lceil \langle V \upharpoonright \alpha \rangle \rceil = (\widetilde{\alpha}\ \lceil V \rceil) \\ \textit{linear abstraction} & \lceil \widetilde{\mu} x.C \rceil = \lambda \widetilde{x}.\lceil C \rceil \quad ; \quad \lceil \mu \alpha.C \rceil = \lambda \widetilde{\alpha}.\lceil C \rceil \\ \textit{linear pair} & \lceil \phi \# \psi \rceil = \langle \lceil \phi \rceil, \lceil \psi \rceil \rangle \quad (\# \in \{\otimes, /, \backslash, \oplus, \oslash, \otimes\}) \\ \textit{case} & \lceil \frac{\phi\ \psi}{\xi}.C \rceil = \mathbf{case}\ \widetilde{\xi}\ \mathbf{of}\ \langle \widetilde{\phi}, \widetilde{\psi} \rangle.\lceil C \rceil \end{array} \tag{36}$$

Sequents For sequent hypotheses/conclusions, we have

$\mathsf{pol}(A)$	$\lceil x : A \rceil$	$\lceil \alpha : A \rceil$
$+$	$\widetilde{x} : \lceil A \rceil$	$\widetilde{\alpha} : \lceil A \rceil^\perp$
$-$	$\widetilde{x} : \lceil A \rceil^\perp$	$\widetilde{\alpha} : \lceil A \rceil$

(37)

Table 1 then specifies how the translation extends to sequents (replace logical connectives by their structural counterparts, and target $\otimes$ by the comma for

multiset union).

$$\begin{array}{rcl} \lceil C : (X \vdash Y) \rceil & = & \lceil X \rceil, \lceil Y \rceil \vdash_{\mathbf{LP}} \lceil C \rceil : \bot \\ \left\lceil X \vdash \boxed{v : A} \right\rceil & = & \lceil X \rceil \vdash_{\mathbf{LP}} \lceil v \rceil : \lceil A \rceil \\ \left\lceil \boxed{e : A} \vdash Y \right\rceil & = & \lceil Y \rceil \vdash_{\mathbf{LP}} \lceil e \rceil : \lceil A \rceil^{\bot} \end{array} \tag{38}$$

Illustrations We return to our sample derivations. In (39) one finds the CPS image of the source types for transitive verb and determiner under the different assignments of bias to the atomic subformulas, and the composition with $\cdot^{\ell}$, assuming $np^{\ell} = e$ (entities) and $s^{\ell} = \bot^{\ell} = t$ (truth values). For the lexical constants of the illustration, Table 2 gives $\cdot^{\ell}$ translations compatible with the typing. In Table 3, these lexical recipes are substituted for the parameters of the CPS translation.

	LG	$\lceil\cdot\rceil^{\bot}$	$(\lceil\cdot\rceil^{\bot})^{\ell}$
a.	$(np^{+}\backslash s^{-})/np^{+}$	$((np \otimes s^{\bot}) \otimes np)^{\bot}$	$((e \times (tt)) \times e) \to t$
b.	np^{+}/n^{+}	$(np^{\bot} \otimes n)^{\bot}$	$((et) \times (et)) \to t$
c.	$(np^{-}\backslash s^{-})/np^{-}$	$((np^{\bot\bot} \otimes s^{\bot}) \otimes np^{\bot\bot})^{\bot}$	$((((et)t) \times (tt)) \times ((et)t)) \to t$
d.	np^{-}/n^{-}	$(np^{\bot} \otimes n^{\bot\bot})^{\bot}$	$((et) \times (((et)t)t)) \to t$

(39)

Table 2. Constants: lexical translations

$(np^{+}\backslash s^{-})/np^{+}$	finds	$\lambda\langle\langle x, c\rangle, y\rangle.(c\ (\textsc{find}^{eet}\ y\ x))$
$(np^{+}/n^{+}) \otimes n^{+}$	everyone	$\langle\lambda\langle x, y\rangle.(\forall\ \lambda z.(\Rightarrow\ (y\ z)\ (x\ z))), \textsc{person}^{et}\rangle$
np^{+}/n^{+}	some	$\lambda\langle x, y\rangle.(\exists\ \lambda z.(\wedge\ (y\ z)\ (x\ z)))$
n^{+}	unicorn	$\textsc{unicorn}^{et}$
$(np^{-}\backslash s^{-})/np^{-}$	needs	$\lambda\langle\langle q, c\rangle, q'\rangle.(q\ \lambda x.(\textsc{need}^{((et)t)et}\ q'\ x))$
$(np^{-}/n^{-}) \otimes n^{-}$	everyone	$\langle\lambda\langle x, w\rangle.(\forall\ \lambda z.(\Rightarrow\ (w\ \lambda y.(y\ z))\ (x\ z))), \lambda k.(k\ \textsc{person}^{et})\rangle$
np^{-}/n^{-}	some	$\lambda\langle x, w\rangle.(\exists\ \lambda z.(\wedge\ (w\ \lambda y.(y\ z))\ (x\ z)))$
n^{-}	unicorn	$\lambda k.(k\ \textsc{unicorn}^{et})$

Table 3. Compositional translations

$$\lceil (33) \rceil = \lambda\widetilde{\beta}.(\mathsf{case}\ \mathsf{subj}^{\ell}\ \mathsf{of}\ \langle \widetilde{y}, \widetilde{z}\rangle.(\mathsf{tv}^{\ell}\ \langle\langle \lambda\widetilde{\gamma}.(\widetilde{y}\ \langle \widetilde{\gamma}, \lambda\widetilde{\gamma}'.(\widetilde{z}\ \widetilde{\gamma}')\rangle), \widetilde{\beta}\rangle, \lambda\widetilde{\alpha}.(\mathsf{det}^{\ell}\ \langle \widetilde{\alpha}, \lambda\widetilde{\alpha}'.(\mathsf{noun}^{\ell}\ \widetilde{\alpha}')\rangle)\rangle)))$$

$$\lceil (33) \rceil^{\ell} = \lambda c.(\forall\ \lambda x.((\Rightarrow\ (\mathsf{person}\ x))\ (c\ ((\textsc{needs}\ \lambda w.(\exists\ \lambda y.((\wedge\ (\mathsf{unicorn}\ y))\ (w\ y))))\ x))))$$

$$\lceil (34) \rceil = \lambda\widetilde{\alpha}.(\mathsf{case}\ \mathsf{subj}^{\ell}\ \mathsf{of}\ \langle \widetilde{x}', \widetilde{z}\rangle.(\widetilde{x}'\ \langle \lambda\widetilde{x}.(\mathsf{det}^{\ell}\ \langle \lambda\widetilde{y}.(\mathsf{tv}^{\ell}\ \langle\langle \widetilde{x}, \widetilde{\alpha}\rangle, \widetilde{y}\rangle), \mathsf{noun}^{\ell}\rangle), \widetilde{z}\rangle))$$

$$\lceil (34) \rceil^{\ell} = \lambda c.(\forall\ \lambda x.((\Rightarrow\ (\textsc{person}\ x))\ (\exists\ \lambda y.((\wedge\ (\textsc{unicorn}\ y))\ (c\ ((\textsc{likes}\ y)\ x))))))$$

$$\lceil (35) \rceil = \lambda\widetilde{\alpha}.(\mathsf{case}\ \mathsf{subj}^{\ell}\ \mathsf{of}\ \langle \widetilde{x}', \widetilde{z}\rangle.(\mathsf{det}^{\ell}\ \langle \lambda\widetilde{y}.(\widetilde{x}'\ \langle \lambda\widetilde{x}.(\mathsf{tv}^{\ell}\ \langle\langle \widetilde{x}, \widetilde{\alpha}\rangle, \widetilde{y}\rangle), \widetilde{z}\rangle), \mathsf{noun}^{\ell}\rangle))$$

$$\lceil (35) \rceil^{\ell} = \lambda c.(\exists\ \lambda y.((\wedge\ (\textsc{unicorn}\ y))\ (\forall\ \lambda x.((\Rightarrow\ (\textsc{person}\ x))\ (c\ ((\textsc{likes}\ y)\ x))))))$$

3.2 Proof nets and focusing

In this section, we introduce term-labeled proof nets, and show how a proof term can be read off from the *composition graph* associated with a net. Our approach is comparable to that of (de Groote and Retoré 1996), who present an algorithm to compute a linear lambda term from a traversal of the dynamic graph associated with a proof net for a derivation in the Lambek calculus. Whereas in the case of the single-conclusion Lambek calculus, the term associated with a given proof net is unique, in the case of multiple-conclusion **LG** there will be the possibility that the term computation algorithm associates more than one term with a proof net. These multiple results will then be shown to correspond to the derivational ambiguity of focused proof search.

Reduction tree When P is a proof net (and therefore converts to a tensor tree using a sequence ρ of conversions and contractions) the components of P can bee seen as a parallel representation of the synchronous phases in sequent proof search. Taking a closer look at the conversion sequence ρ, we see that all interaction rules operate in one component C, the cotensor rules and the corresponding contractions operate on a component to which it is attached by both of its active tentacles (i.e. the tentacles without the arrow) and the contraction removes a tensor link from this component. If the main tentacle points to a vertex attached to a non-trivial component C' then a new component is formed by merging C (minus the contracted tensor link) and C' into a new component. When multiple cotensor links have both active tentacles attached to a single component (Figure 8 shows an example), we can apply all contractions simultaneously: since a contraction connects a tensor and a cotensor link at two out of three tentacles, there cannot be a conflict (multiple cotensor links connected to a tensor link with *both* contractions being impossible without violating the

definition of proof structures). In addition, when the main vertex of a cotensor link is the active vertex of another cotensor link, then, if the other active vertex of this link is connected to the current component as well, we can apply this contraction immediately.

So instead of seeing ρ as a *sequence* of reductions, we can see it as a rooted *tree* of reductions: the initial components are its leaves (synchronous phases) and the contractions connecting multiple components to form new components its are branches (the branches from the active components to their parents correspond to asynchronous phases) and the final tree — a single component — is its root.

Example 5. Figure 15 shows an example of how the view of components given above allows us to see a proof net as a tree of components. The shaded subnet boxes are components and contain only tensor links. For clarity, the cotensor links are shown in the figure as well.

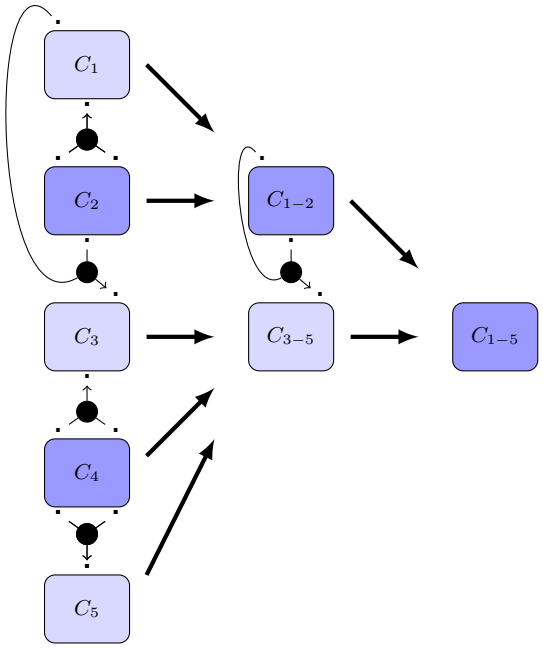


Fig. 15. A reduction sequence seen as a rooted tree

Each interaction rule takes place completely in one of the C_i. In the figure, the components which do not contain the main vertex of a cotensor link are shown

in a darker shade: we will call these components *active*. In Figure 15, C_2 and C_4 are active. Now, it is easy to show that whenever there exists a conversions sequence ρ, we can transform it into a conversion sequence ρ' where conversions take place only in the active components: any conversions in C_1 can be delayed until after the contraction connecting C_1 and C_2, since only C_2 is relevant for this contraction (it contains both active vertices of the cotensor link and therefore also the tensor link it contracts with), and any conversions in C_5 can be delayed until the final component C_{1-5}.

In addition, the two active components C_2 and C_4 are independent: we can apply conversions to these two components in parallel.

Nets and term labeling When assigning a term label to a proof net, we will be interested in assigning labels to larger and larger subnets of a given proof net, until we have computed a term for the complete proof net. Like in the sequent calculus, we distinguish between subnets which are commands, contexts and values. Figure 16 shows how we will distinguish these visually: the main formula of a subnet is drawn white, other formulas are drawn in light gray, values are drawn inside a rectangle, contexts inside an oval.

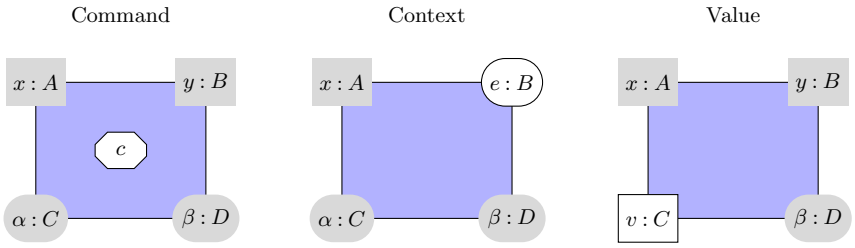


Fig. 16. Proof nets with term labels: commands, context and values

Figure 17 gives the term-labeled version of the proof net links corresponding to the logical rules of the sequent calculus. The flow of information is shown by the arrows: information flow is always from the active formulas to the main formula of a link, and as a consequence the complex term can be assigned either to a conclusion or to a premiss of the link. This is the crucial difference with term labeling for the single-conclusion Lambek calculus, where the complex term is always assigned to a conclusion. The cotensor rules, operating on commands, indicate the prefix for the command corresponding to the term assignment for the rule (we will see later how commands are formed).

The proof term of an **LG** derivation is computed on the basis of the *composition graph* associated with its proof net.

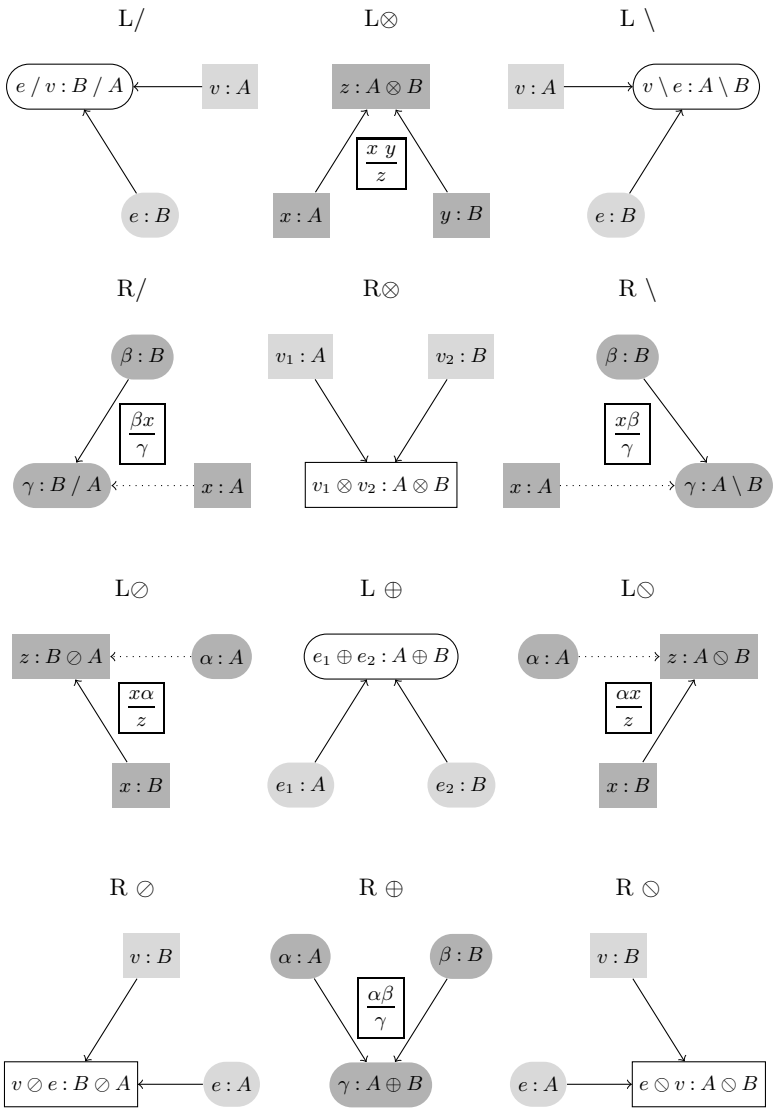


Fig. 17. LG links with term labeling

Definition 8. *Given a proof net P, the associated* composition graph *$cg(P)$ is obtained as follows.*

1. *all vertices of P with formula label A are expanded into* axiom links*: edges connecting two vertices with formula label A; all links are replaced by the corresponding links of Figure 17;*

2. *all vertices in this new structure are assigned atomic terms of the correct type (variable or covariable) and the terms for the tensor rules are propagated from the active formulas to the main formula;*
3. *all axiom links connecting terms of the same type (value or context) are collapsed.*

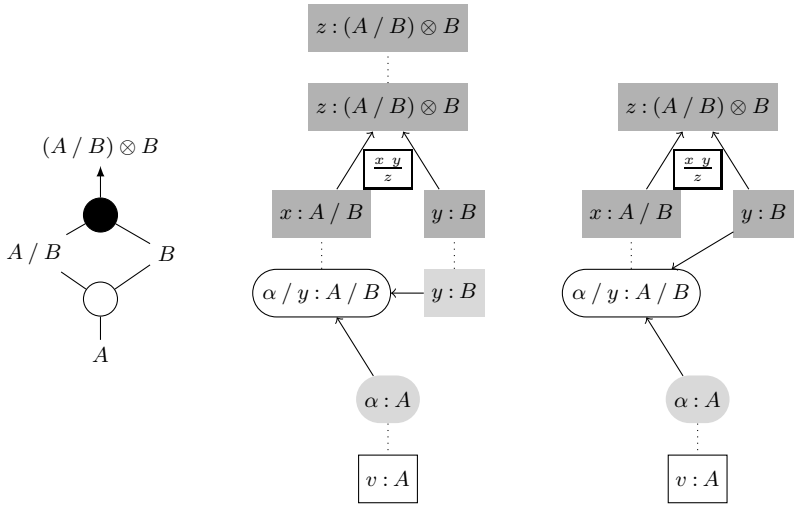


Fig. 18. Proof net (left) and its associated composition graph; in the middle, the expanded net with term annotations; on the right the result of contracting the substitution links.

Figure 18 gives an example of the composition graph associated with a net. In all, the expansion stage gives rise to four types of axiom links, depending on the type of the term assigned to the A premiss and the A conclusion. These cases are summarized in Figure 19. The substitution links are collapsed in the final stage of the construction of the composition graph; the command and $\mu/\widetilde{\mu}$ cases are the ones that remain.

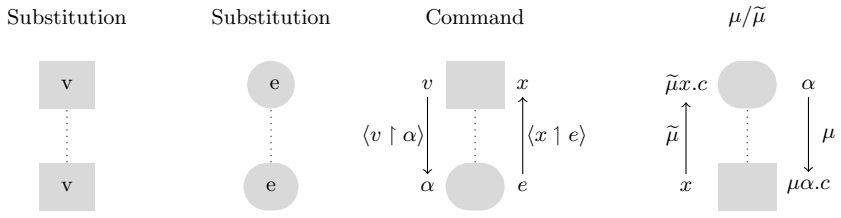


Fig. 19. Types of axiom links

Given the composition graph $cg(P)$ associated with a proof net P, we compute terms for it as follows.

1. we compute all maximal subnets of $cg(P)$, which consist of a set of tensor links with a single main formula, marking all these links as visited;
2. while $cg(P)$ contains unvisited links do the following:
   (a) follow an unvisited command link attached to a previously calculated maximal subnet, forming a correct command subnet; like before, we restrict to *active* subnets which do not contain (or allow us to reach through an axiom) the main formula of a negative link;
   (b) for each negative link with both active formulas attached to the current command subnet, pass to the main formula of the negative link, forming a new command, repeat this step until no such negative links remain attached;
   (c) follow a μ or $\widetilde{\mu}$ link to a new vertex, forming a larger value or context subnet and replacing the variable previously assigned to the newly visited vertex by the μ value or $\widetilde{\mu}$ context.

The algorithm stays quite close to the focused proof nets of the previous section: the maximal subnets of step 1 are *rooted* versions of the components we have used before, with the directions of the arrows potentially splitting components into multiple rooted components (Figure 21 will give an example) and the asynchronous phases, which consisted of one or more contractions for cotensor links, will now consist of a passage through a command link, followed by zero or more cotensor links, followed by either a μ or a $\widetilde{\mu}$ link, the result being a new, larger subnet. The term assignment algorithm is a way to enumerate the non-equivalent proof terms of a net. Given that these terms are isomorphic to focused sequent proofs, it is no coincidence that the computation of the proof terms looks a lot like the sequentialisation algorithm.[4] The following lemma is easy to prove.

Lemma 1. *If P is a proof net (with a pairing of command and $\mu/\widetilde{\mu}$ links) and v is a term calculated for P using this pairing then there is a sequent proof π which is assigned v as well.*

This lemma is easily proved by induction on the depth of the tree: it holds trivially for the leaves (which are rooted components), and, inductively, each command, contensor, $\mu/\widetilde{\mu}$ sequence will produce a sequent proof of the same term: in fact each such step corresponds exactly to the derived inference rules for focus shifting discussed in §3.1.

To summarize: the difference between computing terms for proof nets in the Lambek calculus and in **LG** can be characterized by the following statements:

[4] The connection between proof net sequentialisation and focusing for linear logic is explored in (Andreoli and Maieli 1999)

Lambek calculus: the (potential) terms are given through a bijection between premiss and conclusion atomic formulas (ie. a complete matching of the axioms),

LG: the (potential) terms are given through a bijection between premiss and conclusion atomic formulas *plus* a bijection between command and $\mu/\widetilde{\mu}$ axioms.

We speak of *potential* terms, since in the case of the Lambek calculus only proof nets can be assigned a term, whereas in the **LG** case we need proof nets plus a coherent bijection between command and $\mu/\widetilde{\mu}$ axioms, where the μ or $\widetilde{\mu}$ rule is applied to one of the free variables of the command c.

Illustrations Figure 20 shows how to compute the term for the example proof net of Figure 18, starting from the composition graph (on the right). We first look for the components (step 1). Since there is only a single tensor link, this is simple. Figure 20 shows, on the left, the context subnet corresponding to this link.

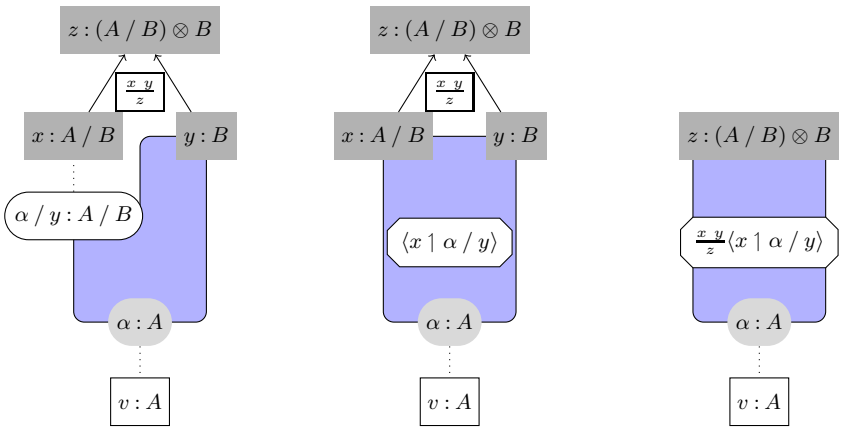


Fig. 20. Computing the proof term from a composition graph

Now, there is only one command to follow from here (step 2a), which produces the command shown in the middle of Figure 20. Applying the cotensor link (step 2b) produces the figure shown on the right. The final μ link (step 2c, not shown) produces the completed term for this proof net.

$$v = \mu\alpha.\frac{x\ y}{z}\langle x \restriction \alpha \,/\, y \rangle$$

Some remarks about this example. First, some of the axioms can be traversed in only one of the two possible directions: in cut-free proof nets, command links

move either towards the active formulas of cotensor links or towards "dead ends": hypotheses or conclusions of the proof net. And since we want to compute the value of v for the example proof net, it only makes sense to apply a μ rule to compute this value: we always "exit" the proof net from a designated conclusion. With a slight modification to the algorithm that reads off terms from a composition graph, we could also compute *commands* for proof nets, or compute the *context* for a designated *premiss* of the net.

Figure 21 returns to our "subj tv det noun" example. On the left we see the composition graph for the example of Figure 11.

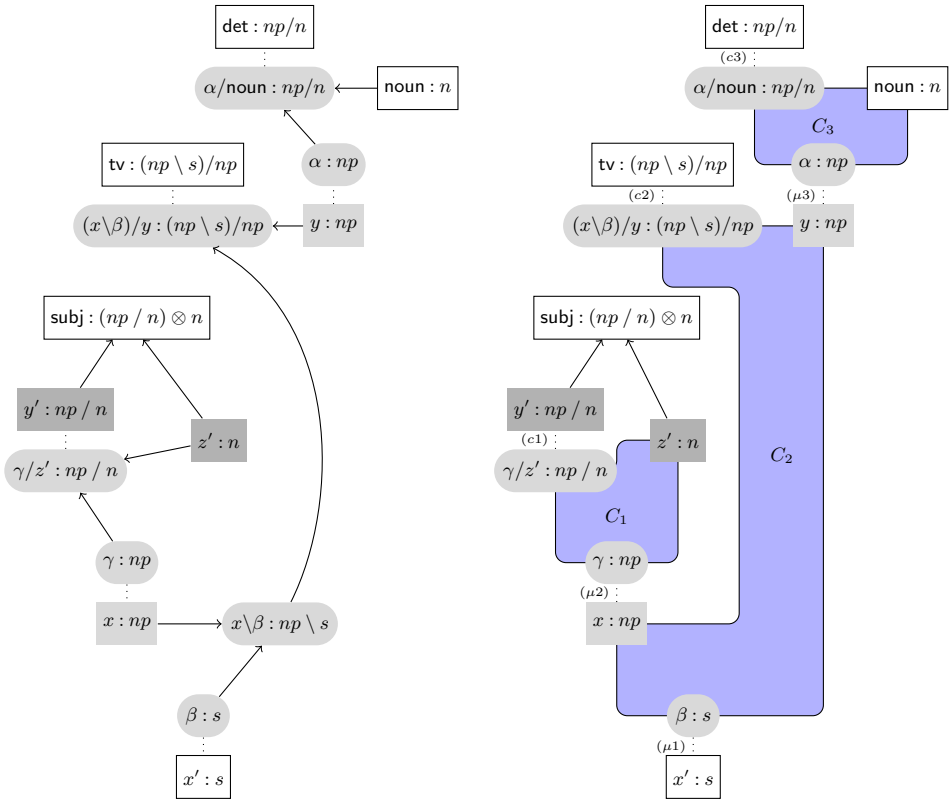


Fig. 21. Composition graph (left) and initial components (right) for the "subj tv det noun" example

The only cotensor link in the figure has the node subj : $(np \,/\, n) \otimes n$ as its main formula. When we compute the rooted components, we see that there are three, shown on the right of the figure. There are three command axioms, one for the root node of each of the three components, C_1 to C_3 on the right hand side of the figure; these are numbered $c1$ to $c3$ next to the corresponding links with the

same number as the corresponding component. There are also three $\mu/\widetilde{\mu}$ links (numbered $\mu 1$ to $\mu 3$). Figure 22 gives a schematic representation of the proof net of Figure 21.

Since we are interested in calculating the value of x', ($\mu 1$) will be the last link we pass in the proof net and therefore we will pass it downwards, producing a term of the form $\mu\beta.c$. Figure 22 gives a schematic representation of the proof net of Figure 21. The arrows next to the $\mu/\widetilde{\mu}$ links indicate the different possibilities for traversing the link and whether this traversal corresponds to a μ or a $\widetilde{\mu}$ link.

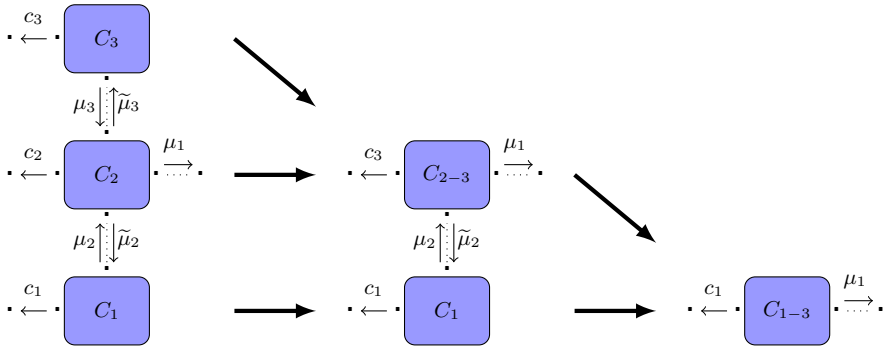


Fig. 22. Matching: $c_2 - \widetilde{\mu}_3$, $c_3 - \widetilde{\mu}_2$, $c_2 - \mu_1$. Reading: $\mathsf{subj} < \mathsf{det} < \mathsf{tv}$.

If both *np* arguments of the transitive verbs are lexically assigned a positive bias, then we can only pass the two axioms $\mu_2/\widetilde{\mu}_2$ and $\mu_3/\widetilde{\mu}_3$ in the $\widetilde{\mu}_2$ and $\widetilde{\mu}_3$ directions, following the arrows away from component C_2. This will necessarily mean that the first command is c_2 and that we can follow this command either with $\widetilde{\mu}_2$ (going to the component of the subject and producing the narrow scope reading for the subject quantifier) or with $\widetilde{\mu}_3$ (going to the component of the object determiner and producing the narrow scope reading for the object quantifier). The figure shows (in the middle) the result of choosing $c_2 - \widetilde{\mu}_3$.

The term computed for component C_2 by following command c_2 is $\langle \mathsf{tv} \uparrow (x\backslash\beta)/y\rangle$ and the $\widetilde{\mu}_3$ link joins components C_2 and C_3, replacing covariable α by the complex context $\widetilde{\mu}y.\langle \mathsf{tv} \uparrow (x\backslash\beta)/y\rangle$, producing the configuration shown schematically in the middle of Figure 22 (refer back to Figure 21 to see the initial labels).

From this middle configuration and given the restriction to $\widetilde{\mu}_2$ for the link connecting the two remaining components, only the command c_3 is a possible in combination with the $\widetilde{\mu}_2$ link. Command c_3 would produce $\langle \mathsf{det} \uparrow \alpha/\mathsf{noun}\rangle$ from the configuration shown in Figure 21 but given the previous substitution for α it will now produce $\langle \mathsf{det} \uparrow (\widetilde{\mu}y.\langle \mathsf{tv} \uparrow (x\backslash\beta)/y\rangle)/\mathsf{noun}\rangle$ and the $\widetilde{\mu}_2$ link will replace covariable γ by the context $\widetilde{\mu}x.\langle \mathsf{det} \uparrow (\widetilde{\mu}y.\langle \mathsf{tv} \uparrow (x\backslash\beta)/y\rangle)/\mathsf{noun}\rangle$ and produce the configuration shown schematically on the right of Figure 22.

The final command c_1 produces $\langle y' \restriction (\widetilde{\mu}x.\langle \mathsf{det} \restriction (\widetilde{\mu}y.\langle \mathsf{tv} \restriction (x\backslash\beta)/y\rangle)/\mathsf{noun}\rangle)/z'\rangle$, the cotensor link the extended command

$$\frac{y'z'}{\mathsf{subj}}.\langle y' \restriction (\widetilde{\mu}x.\langle \mathsf{det} \restriction (\widetilde{\mu}y.\langle \mathsf{tv} \restriction (x\backslash\beta)/y\rangle)/\mathsf{noun}\rangle)/z'\rangle \tag{40}$$

and, finally, by μ_1 the term for the complete proof net (and its command-$\mu/\widetilde{\mu}$ pairing):

$$\mu\gamma.\frac{y'z'}{\mathsf{subj}}.\langle y' \restriction (\widetilde{\mu}x.\langle \mathsf{det} \restriction (\widetilde{\mu}y.\langle \mathsf{tv} \restriction (x\backslash\beta)/y\rangle)/\mathsf{noun}\rangle)/z'\rangle \tag{41}$$

Similarly, starting with the $c_2 - \widetilde{\mu}_2$ pairing will produce.

$$\mu\beta.\langle \mathsf{det} \restriction (\widetilde{\mu}y.\frac{y'z'}{\mathsf{subj}}.\langle y' \restriction (\widetilde{\mu}x.\langle \mathsf{tv} \restriction (x\backslash\beta)/y\rangle)/z'\rangle)/\mathsf{noun}\rangle \tag{42}$$

These are the only two readings available with positive bias for the two atomic *np* arguments of the transitive verb, and, as we have seen before, this gives the right quantifier scope possibilities for an extensional transitive verb such as "likes" we have seen in equations (34) and (35) (apart from the variable names, equation (42) differs from (35) in that the extended command fraction in the latter term is at the innermost position, but the terms are equivalent up to commutative conversions).

When we use a negative bias for the two *np* arguments of the transitive verb, we obtain the following term, corresponding to equation (33).

$$\mu\beta.\frac{y'z'}{\mathsf{subj}}.\langle y' \restriction (\widetilde{\mu}x.\langle \mathsf{tv} \restriction (x\backslash\beta)/(\mu\alpha.\langle \mathsf{det} \restriction \alpha/\mathsf{noun}\rangle)\rangle)/z'\rangle \tag{43}$$

4 Conclusions

The Lambek-Grishin calculus is a symmetric version of the Lambek calculus. Together with the interaction principles, it allows for the treatment of patterns beyond context-free which cannot be satisfactorily handled in the Lambek calculus. We have compared two proof systems for **LG**: focused sequent proofs and proof nets. Focused proofs avoid the spurious non-determinism of backward-chaining search in the sequent calculus; they provide a natural interface to semantic interpretation via their continuation-passing-style translation. Proof nets present the essence of a derivation in a visually appealing form; they do away with the syntactic clutter of sequent proofs, and compute the structure of the end-sequent in a data-driven manner where this structure has to be given before one can a start backward-chaining sequent derivation. Proof terms are read off from the composition graph associated with a net. The computation of these terms depends both on a bijection between premise and conclusion atomic formulas and between command and $\mu/\widetilde{\mu}$ axioms. As a result, one net can be associated with multiple construction recipes (proof terms), corresponding to multiple derivations in the focused sequent calculus.

Table of Contents

References

Andreoli, J.-M. (2001). Focussing and proof construction. *Annals of Pure and Applied Logic 107*(1-3), 131–163.

Andreoli, J.-M. and R. Maieli (1999). Focusing and proof-nets in linear and non-commutative logic. In *International Conference on Logic for Programming and Automated Reasoning (LPAR)*, Volume 1581 of *LNAI*. Springer.

Bastenhof, A. (2011). Polarized Montagovian semantics for the Lambek-Grishin calculus. *CoRR abs/1101.5757*. To appear in the Proceedings of the 15th Conference on Formal Grammar (Copenhagen, 2010), Springer LNCS.

Bernardi, R. and M. Moortgat (2007). Continuation semantics for symmetric categorial grammar. In D. Leivant and R. de Queiros (Eds.), *Proceedings 14th Workshop on Logic, Language, Information and Computation (WoLLIC'07)*, LNCS 4576. Springer.

Bernardi, R. and M. Moortgat (2010). Continuation semantics for the Lambek-Grishin calculus. *Information and Computation 208*(5), 397–416.

Cockett, J. and R. Seely (1996). Proof theory for full intuitionistic linear logic, bilinear logic and mix categories. In *Theory and Applications of Categories 3*, pp. 85–131.

Curien, P. and H. Herbelin (2000). Duality of computation. In *International Conference on Functional Programming (ICFP'00)*, pp. 233–243. [2005: corrected version].

de Groote, P. and C. Retoré (1996). Semantic readings of proof nets. In G.-J. Kruijff, G. Morrill, and D. Oehrle (Eds.), *Formal Grammar*, Prague, pp. 57–70. FoLLI.

Girard, J.-Y. (1987). Linear logic. *Theoretical Computer Science 50*, 1–102.

Girard, J.-Y. (1991). A new constructive logic: classical logic. *Mathematical Structures in Computer Science 1*(3), 255–296.

Goré, R. (1997). Substructural logics on display. *Logic Journal of IGPL 6*(3), 451–504.

Grishin, V. (1983). On a generalization of the Ajdukiewicz-Lambek system. In A. Mikhailov (Ed.), *Studies in Nonclassical Logics and Formal Systems*, pp. 315–334. Moscow: Nauka. [English translation in Abrusci and Casadio (eds.) Proceedings 5th Roma Workshop, Bulzoni Editore, Roma, 2002].

Kallmeyer, L. (2010). *Parsing Beyond Context-Free Grammars*. Cognitive Technologies. Springer.

Lambek, J. (1958). The mathematics of sentence structure. *American Mathematical Monthly 65*, 154–170.

Lambek, J. (1961). On the calculus of syntactic types. In R. Jacobson (Ed.), *Structure of Language and its Mathematical Aspects, Proceedings of the Symposia in Applied Mathematics*, Volume XII, pp. 166–178. American Mathematical Society.

Lambek, J. (1988). Categorial and categorical grammars. In E. B. R. Oehrle and D. Wheeler (Eds.), *Categorial Grammars and Natural Language Structures*, pp. 297–317. Dordrecht: Reidel Publishing Company.

Lambek, J. (1993). From categorial to bilinear logic. In K. D. P. Schröder-Heister (Ed.), *Substructural Logics*, pp. 207–237. Oxford University Press.

Moortgat, M. (1996). Multimodal linguistic inference. *Journal of Logic, Language and Information 5*(3,4), 349–385.

Moortgat, M. (2007). Symmetries in natural language syntax and semantics: the Lambek-Grishin calculus. In *Proceedings of WoLLIC 2007*, Volume 4567 of *LNCS*, pp. 264–284. Springer.

Moortgat, M. (2009). Symmetric categorial grammar. *Journal of Philosophical Logic 38*(6), 681–710.

Moot, R. (2007). Proof nets for display logic. Technical report, CNRS and INRIA Futurs.

Moot, R. and Q. Puite (2002). Proof nets for the multimodal Lambek calculus. *Studia Logica 71*(3), 415–442.

Morrill, G. (1994). *Type Logical Grammar*. Dordrecht:Kluwer.

Morrill, G., M. Fadda, and O. Valentin (2007). Nondeterministic discontinuous Lambek calculus. In *Proceedings of the Seventh International Workshop on Computational Semantics (IWCS7)*, Tilburg.